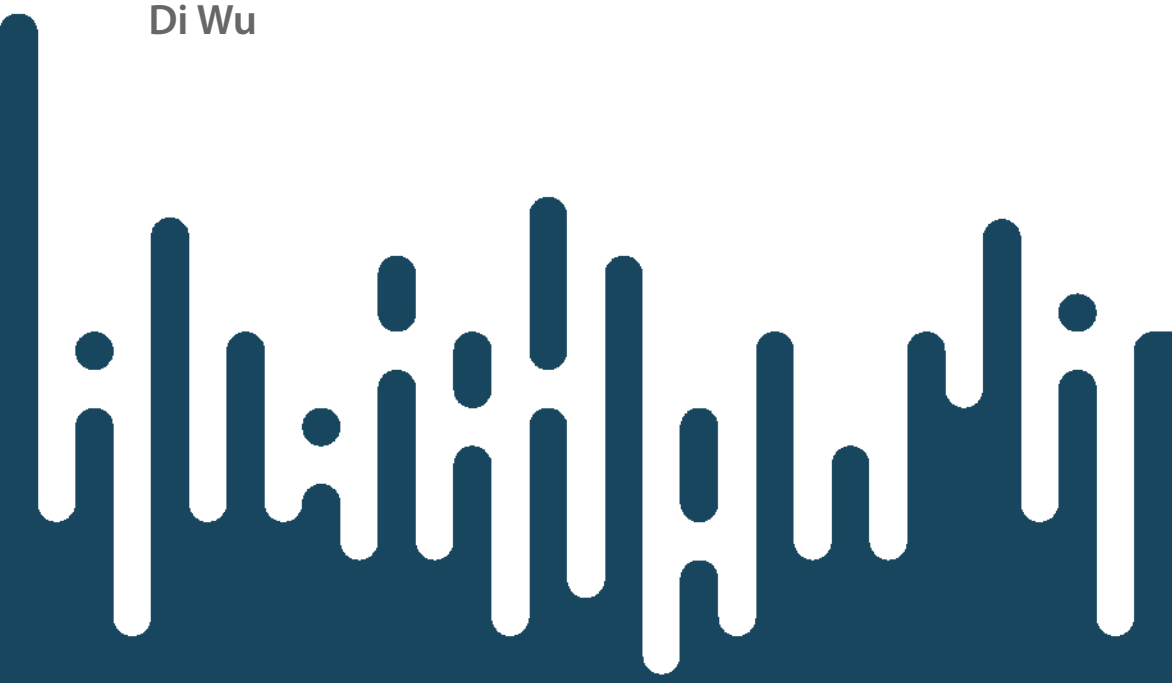
SpringerBriefs in Computer Science

Di Wu



Robust Latent Feature Learning for Incomplete Big Data

Springer

SpringerBriefs in Computer Science

SpringerBriefs present concise summaries of cutting-edge research and practical applications across a wide spectrum of fields. Featuring compact volumes of 50 to 125 pages, the series covers a range of content from professional to academic.

Typical topics might include:

- A timely report of state-of-the art analytical techniques
- A bridge between new research results, as published in journal articles, and a contextual literature review
- A snapshot of a hot or emerging topic
- An in-depth case study or clinical example
- A presentation of core concepts that students must understand in order to make independent contributions

Briefs allow authors to present their ideas and readers to absorb them with minimal time investment. Briefs will be published as part of Springer's eBook collection, with millions of users worldwide. In addition, Briefs will be available for individual print and electronic purchase. Briefs are characterized by fast, global electronic dissemination, standard publishing contracts, easy-to-use manuscript preparation and formatting guidelines, and expedited production schedules. We aim for publication 8–12 weeks after acceptance. Both solicited and unsolicited manuscripts are considered for publication in this series.

**Indexing: This series is indexed in Scopus, Ei-Compendex, and zbMATH **

Di Wu

Robust Latent Feature Learning for Incomplete Big Data

Springer

Di Wu
College of Computer and Information Science
Southwest University
Chongqing, China

ISSN 2191-5768 ISSN 2191-5776 (electronic)
SpringerBriefs in Computer Science
ISBN 978-981-19-8139-5 ISBN 978-981-19-8140-1 (eBook)
https://doi.org/10.1007/978-981-19-8140-1

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Preface

In the era of Big Data, high dimensional and incomplete (HDI) data are frequently encountered in many industrial applications, such as recommender systems, the Internet of Things, intelligent transportation, cloud computing, and so on. It is of great significance to analyze them for mining rich and valuable knowledge and patterns. Latent feature learning is one of the most popular representation learning methods tailored for HDI data due to its high accuracy, computational efficiency, and ease of scalability. The crux of analyzing HDI data lies in addressing the uncertainty problem caused by their incomplete characteristics. However, existing HDI methods do not fully consider such uncertainty.

In this book, I introduce several robust latent feature learning methods to address such uncertainty for effectively and efficiently analyzing HDI data, including robust latent feature learning based on smooth L_1-norm, improving robustness of latent feature learning using L_1-norm, improving robustness of latent feature learning using double-space, data-characteristic-aware latent feature learning, posterior-neighborhood-regularized latent feature learning, and generalized deep latent feature learning.

This is the first book that can help researchers and engineers fully understand how to employ robust latent feature learning to effectively and efficiently analyze HDI data. It is assumed that readers have a basic knowledge of mathematics, as well as a certain background in data mining. Readers can obtain an overview of the challenges of analyzing HDI data. In addition, this book provides several algorithms and real application cases, which can help readers quickly build their robust latent feature learning models to analyze HDI data.

Chongqing, China — Di Wu
September 2022

Acknowledgments

The work described in this book was supported by the National Natural Science Foundation of China under grant 62176070.

Contents

About the Author

Di Wu received a Ph.D. degree in Computer Science from Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China, in 2019. He was a visiting scholar from April 2018 to April 2019 at the University of Louisiana, Lafayette, USA. Currently, he is a Professor at the College of Computer and Information Science, Southwest University. His current research interests include data mining, artificial intelligence, and big data. He has published over 50 papers, including 12 IEEE TRANSACTIONS papers, three highly cited paper of ESI, and several top-tier conferences such as AAAI, ICDM, WWW, and IJCAI. His Google Scholar citations are more than 1800, and his H-Index is 23. He is an Associate Editor for Frontiers in Neurorobotics (SCI, IF 3.493). He received the Nomination Award for Excellent Doctoral Dissertation of the Chinese Association for Artificial Intelligence (CAAI).

Chapter 1
Introduction

1.1 Background

In the era of Big Data, information explosion is very common in our daily life [1–4]. For instance, Google and Flickr generate more than 20 PB and 3.6 TB data per day, respectively [5]. According to the prediction of the International Data Corporation, the global data sum will go to 175 ZB by 2025 [6]. Therefore, how to effectively and efficiently mine the desired valuable information from large-scale data has become a crucial challenge and has attracted much attention all over the world [7–11].

In many real applications such as social media, healthcare, electronic commerce, bioinformatics, smart grid, online education, and intelligent transportation, the features of big data are continuously generated over time [12–15], making the collected data from these applications have a typical high dimensionality characteristic [16–18]. High dimensionality can cause many serious problems [19, 20], including high computational and storage costs, performance degradation on unseen data, comprehension and visualization difficulties, etc. Thus, it is necessary to reduce the original high-dimensional data into a low-dimensional feature space [20, 21] while keeping their essential discriminative features [22, 23].

Besides high dimensionality, incompletion is another typical characteristic of big data in many real applications [24–26]. For example, the collected data from wireless sensor networks are commonly incomplete due to saving energy by deliberately shutting down partial sensors, human errors, sudden sensor breakdown, etc. Another example is recommender systems, where the user-item interaction relationships cannot be completely observed because each user cannot touch all the numerous items in practice [27–29]. The incompletion of data brings serious difficulties to data analysis as missing data/values result in many uncertainties. Thus, it is required to develop an effective approach to handle the incomplete issues for big data analysis [30].

D. Wu, *Robust Latent Feature Learning for Incomplete Big Data*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-19-8140-1_1

As discussed above, high dimensionality and incompletion are two inherent characteristics of big data. The crux of mining the desired valuable information from big data lies in precisely representing their two inherent characteristics. In this book, the big data with such two inherent characteristics are termed High Dimensional and Incomplete (HDI) data.

To date, various approaches have been proposed to represent the HDI data. Among them, Latent Feature Learning (LFL) is one of the most popular models [30, 31]. It has proven to be highly effective and efficient in representing an HDI matrix. Its principle is to learn two low-rank latent feature matrices to represent the original HDI matrix. It is trained by minimizing the differences between the only known data of HDI matrix and the corresponding predictions relying on the two latent feature matrices.

Notably, HDI data contain many uncertainties caused by missing data/values. Besides, some outliers (e.g., noises) inevitably mix in the HDI data to further aggravate such uncertainties. Hence, an LFL model is required to be much more robust in representing the HDI data. By investing in previous studies, however, no one systemically discusses the robust LFL approaches in representing the HDI data. To fill this gap, this book introduces several robust LFL models to address the uncertainties for effectively and efficiently representing the HDI data. Besides, it provides several algorithms and real application cases, which can help students, researchers, and professionals quickly build their models to analyze HDI data.

1.2 Symbols and Notations (Table 1.1)

Table 1.1 The adopted symbols and notations

Notation	Explanation
HDI	The abbreviation of high dimensional and incomplete.
M,N	Two types of entity sets.
m,n	Single entity of M and N, respectively. $m \in M$, $n \in N$.
$\mathbf{H}$	Targeted HDI matrix with $\|M\|$ rows and $\|N\|$ columns.
$h_{m,n}$	An entry of $\mathbf{H}$ at m-th row and n-th column.
H_{O}, H_{U}	Observed and unobserved entry sets of $\mathbf{H}$.
$\boldsymbol{\Omega}$	A binary index matrix with $\|M\|$ rows and $\|N\|$ columns.
$\Omega_{m,n}$	An entry of $\boldsymbol{\Omega}$ at m-th row and n-th column *w.r.t.* the observation situation of $h_{m,n}$.
d	Latent feature dimension.
$\mathbf{X}$	A latent feature matrix *w.r.t.* M with $\|M\|$ rows and d columns.
$\mathbf{Y}$	A latent feature matrix *w.r.t.* N with $\|N\|$ rows and d columns.
$x_{m,j}$	An entry of $\mathbf{X}$ at m-th row and j-th column, $j \in \{1,2, \ldots, d\}$.
$y_{n,j}$	An entry of $\mathbf{Y}$ at n-th row and j-th column, $j \in \{1,2, \ldots, d\}$.
$\hat{\mathbf{H}}$	$\mathbf{H}$'s rank-d approximation matrix with $\|M\|$ rows and $\|N\|$ columns.
$\hat{H}_{m,n}$	An entry of $\hat{\mathbf{H}}$ at m-th row and n-th column. It is the prediction of $h_{m,n}$.
$\Delta_{m,n}$	Prediction error between $h_{m,n}$ and $\hat{H}_{m,n}$, i.e., $\Delta_{m,n} = h_{m,n} - \hat{H}_{m,n}$.

(continued)

Table 1.1 (continued)

Notation	Explanation
$\mathbf{W}$	$\mathbf{H}$'s linear bias matrix with $\|M\|$ rows and $\|N\|$ columns.
$w_{m,n}$	An entry of $\mathbf{W}$ at m-th row and n-th column. It is the linear bias of $h_{m,n}$.
μ	Global average value of H_O,
$\mathbf{b_M}$	Linear bias vector *w.r.t.* M with size of $\|M\|$.
b_m	The m-th element of $\mathbf{b_M}$ denoting the linear bias *w.r.t.* m-th entity of M.
$\mathbf{b_N}$	Linear bias vector *w.r.t.* N with size of $\|N\|$.
b_n	The n-th element of $\mathbf{b_N}$ denoting the linear bias *w.r.t.* n-th entity of N.
λ	A hyper-parameter of regularization coefficient.
η	A hyper-parameter of learning rate.
t,t_{max}	t-th training iteration, $t\in\{1, 2, \ldots, t_{max}\}$, t_{max} *is* maximum.
Γ	Testing set.
$\|\cdot\|_F$	The Frobenius norm of a matrix.
$\|\cdot\|_{L2}$	The L_2-norm of a matrix.
$\|\cdot\|_{L1}$	The L_1-norm of a matrix.
$\|\cdot\|_{SL1}$	The smooth L_1-norm of a matrix.
$\odot$	The Hadamard product (component-wise multiplication) of two matrices.

1.3 Book Organization

This book is organized as follows:

Chapter 1: This chapter provides an overview of robust LFL for incomplete big data and some background knowledge. Besides, the adopted main symbols and notations are summarized.

Chap. 2: This chapter introduces the basis of LFL. First, it introduces the principle of LFL. Then, two LFL models are introduced. Finally, some experiments are conducted to evaluate the performance of the two LFL models.

Chap. 3: This chapter introduces a smooth L1-norm-oriented latent feature (SL-LF) model to predict the missing data of an HDI matrix robustly and accurately. A large number of experiments on eight HDI matrices are carried out to evaluate the SL-LF model. The experimental results verify the effectiveness of the SL-LF model.

Chap. 4: This chapter introduces an L_1 and L_2 norm-oriented Latent feature (L3F) model. A large number of experiments on nine HDI matrices from recommender systems are carried out to evaluate the L3F model. The results demonstrate that the L3F model outperforms its competitors.

Chap. 5: This chapter introduces a double-space and double-norm ensembled latent feature (D2E-LF) model. The D2E-LF model is evaluated on two HDI matrices from the Web service. The experimental results show that the D2E-LF model has a better Quality-of-Service (QoS) prediction accuracy than its competitors.

Chap. 6: This chapter introduces a data-characteristic-aware latent feature (DCALF) model. The DCALF model is evaluated on two HDI matrices from the

Web service. The experimental results show that the DCALF model has a better QoS prediction accuracy than its competitors.

Chap. 7: This chapter introduces a posterior-neighborhood-regularized latent feature (PLF) model. The PLF model is evaluated on an HDI matrix from the Web service. The experimental results show that the PLF model has a better QoS prediction accuracy than its competitors.

Chap. 8: This chapter introduces a deep latent feature (DLF) model. The DLF model is evaluated on four HDI matrices from recommender systems. The experimental results substantiate that the DLF model has a better missing data prediction accuracy than its competitors.

Chap. 9: This chapter concludes this book and provides some future directions that can be explored.

Notably, to make each chapter self-contained, some crucial contents, e.g., motivations, model definitions, and datasets appearing in previous chapters, may be briefly reiterated in some later chapters.

References

1. Xindong, W., Zhu, X., Gong-Qing, W., Ding, W.: Data mining with big data. IEEE Trans. Knowl. Data Eng. **26**(1), 97–107 (2014)
2. Zhang, S., Yao, L., Sun, A.: Deep learning based recommender system: a survey and new perspectives. ACM Comput. Surv. **51**(1), 1–35 (2019)
3. Lu, R., Jin, X., Zhang, S., Qiu, M., Wu, X.: A study on big knowledge and its engineering issues. IEEE Trans. Knowl. Data Eng. **31**(9), 1630–1644 (2019)
4. Zhang, Q., Yang, L.T., Chen, Z., Li, P.: A survey on deep learning for big data. Information Fusion. **42**, 146–157 (2018)
5. Patrizio, A.: Idc: expect 175 zettabytes of data worldwide by 2025. Network World from IDC. (2018). https://www.networkworld.com/article/3325397/idc-expect-175-zettabytes-of-data-worldwide-by-2025.html
6. Athey, S.: Beyond prediction: using big data for policy problems. Science. **355**(6324), 483–485 (2017)
7. Di, W., Shang, M., Luo, X., Ji, X., Yan, H., Deng, W., Wang, G.: Self-training semi-supervised classification based on density peaks of data. Neurocomputing. **275**, 180–191 (2018)
8. D. Wu, M. Shang, G. Wang, and L. Li. A self-training semi-supervised classification algorithm based on density peaks of data and differential evolution. 2018 IEEE 15th international conference on networking, Sensing and Control (ICNSC), 1-6, (2018)
9. Xuegang, H., Zhou, P., Li, P., Wang, J., Xindong, W.: A survey on online feature selection with streaming features. Front. Comp. Sci. **12**(3), 479–493 (2018)
10. Z. Yu, D. Wu, and Y. He. A robust latent factor analysis model for incomplete data recovery in wireless sensor networks. 2022 *IEEE International Conference on Edge Computing and Communications (EDGE)*), 178–183, (2022)
11. Chen, J., Wang, R., Wu, D., Luo, X.: A differential evolution-enhanced position-transitional approach to latent factor analysis. IEEE Transactions on Emerging Topics in Computational Intelligence, 1–13 (2022)
12. Deng, S., Chen, F., Di, W., He, Y., Ge, H., Ge, Y.: Quantitative combination load forecasting model based on forecasting error optimization. Comput. Electr. Eng. **101**, 108125 (2022)

13. He, Y., Baijun, W., Di, W., Beyazit, E., Chen, S., Xindong, W.: Online learning from capricious data streams: a generative approach. Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI. **2019**, 2491–2497 (2019)
14. Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R.P., Tang, J., Liu, H.: Feature selection: a data perspective. ACM Computing Surveys (CSUR). **50 (6), 94** (2018)
15. Xiaoyu, X., Pang, G., Di, W., Shang, M.: Joint hyperbolic and euclidean geometry contrastive graph neural networks. Inf. Sci. **609**, 799–815 (2022)
16. Alelyani, S., Tang, J., Liu, H.: Feature Selection for Clustering: a Review': 'Data Clustering, pp. 29–60. Chapman and Hall/CRC (2018)
17. Chandrashekar, G., Sahin, F.: A survey on feature selection methods. Computers & Electrical Engineering. **40**(1), 16–28 (2014)
18. Jiliang Tang, Salem Alelyani, and Huan Liu: 'Feature selection for classification: A review': 'Data classification: Algorithms and applications' (2014), pp. 37–64
19. Xindong, W., Kui, Y., Ding, W., Wang, H., Zhu, X.: Online feature selection with streaming features. IEEE Trans. Pattern Anal. Mach. Intell. **35**(5), 1178–1192 (2013)
20. Ege Beyazit, Jeevithan Alagurajah, and Xindong Wu. Online learning from data streams with varying feature spaces. *In Proceedings of the 33rd AAAI Conference on Artificial Intelligence*), 3232–3239, (2019)
21. Xiaoyu, X., Di, W., Shang, M.: A structure-characteristic-aware network embedding model via differential evolution. Expert Syst. Appl. **204**, 117611 (2022)
22. Wu, L., Sun, P., Hong, R., Ge, Y., Wang, M.: Collaborative neural social recommendation. IEEE Trans. Syst. Man Cybern. **Systems**, 1–13 (2018)
23. Antoine Boutet, Davide Frey, Rachid Guerraoui, Arnaud Jegou, and Anne Marie Kermarrec. Whatsup: A decentralized instant news recommender. *In 27th International Parallel and Distributed Processing Symposium (IPDPS)*), 741–752, (2013)
24. Steffen Rendle, Walid Krichene, Li Zhang, and John R. Anderson. Neural collaborative filtering vs. Matrix factorization revisited. *In Proceedings of the 14th ACM Conference on Recommender Systems, Rec Sys*, 240–248, (2020)
25. Hong-Jian Xue, Xin-Yu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. *In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 3203–3209, (2017)
26. Dong, X., Lei, Y., Zhonghuo, W., Sun, Y., Yuan, L., Zhang, F.: A hybrid collaborative filtering model with deep structure for recommender systems. In proceedings of Thirty-First AAAI Conference on Artificial Intelligence. **1309-1315** (2017)
27. Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. *In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. ACM*), 153–162, (2016)
28. Koren, Y., Bell, R.: Advances in Collaborative Filtering': 'Recommender Systems Handbook, pp. 77–118. Springer (2015)
29. Ricci, F., Rokach, L., Shapira, B.: Introduction to Recommender Systems Handbook': 'Recommender Systems Handbook, pp. 1–35. Springer (2011)
30. P. Zhang, Y. He, and D. Wu. An ensemble latent factor model for highly accurate web service qos prediction. 2021 *IEEE International Conference on Big Knowledge (ICBK)*), 361–368, (2021)
31. D. Wu, G. Lu, and Z. Xu. Robust and accurate representation learning for high-dimensional and sparse matrices in recommender systems. 2020 *IEEE International Conference on Knowledge Graph (ICKG)*), 489–496, (2020)

Chapter 2
Basis of Latent Feature Learning

2.1 Overview

The Internet is speeding up the development of information ear. Online service applications have become very common in our daily life and thousands of services are provided online. Such numerous online services lead to the problem of information overload [1, 2]. Then, an intelligent and efficient system is desired to address such problem [3, 4]. Therefore, as one of the most efficient and effective approaches for addressing information load, the recommender system has attracted much attention.

In general, existing recommender systems can be classified into two categories [5–11], i.e., content-based and collaborative filtering (CF) models. Since CF-based models possess the advantages of high computational efficiency and convenient implementation, they have been widely adopted during the past decade [12, 13]. A CF-based model frequently uses a user-item rating matrix to represent users' preferences on items, where higher ratings represent stronger preferences. However, since a user is impossible to touch all the items, making such a user-item rating matrix highly sparse [14]. Hence, the problem of recommendation is to estimate the missing data of the high dimensional and incomplete (HDI) user-item rating matrix [15, 16].

The latent feature learning (LFL)-based model [17–20] is a representative CF-based model. Its principle is to map both items and users into the same latent feature space to make the estimations for the missing ratings of the HDI user-item rating matrix [21–23]. Generally, some products are widely perceived as better (or worse) than others. Thus, an LFL-based recommender frequently adoptes linear biases (LBs) strategy to improve prediction accuracy as well as convergence rate [24, 25]. Next, two basic LFL models in which one considers LBs and the other one does not consider LBs are introduced.

D. Wu, *Robust Latent Feature Learning for Incomplete Big Data*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-19-8140-1_2

2.2 Preliminaries

The LFL problem is to reconstruct an HDI matrix that is defined in Definition 2.1 [26–32]. Then, the LFL is defined in Definition 2.2.

Definition 2.1 HDI matrix. Given entity set M and entity set N, the relationship between M and N can be described by a $|M| \times |N|$ matrix $\mathbf{H}$. Each element $h_{m,n}$ in $\mathbf{H}$ represents the quantitative weight of the relationship between entity $m \in M$ and entity $n \in N$. Let $\mathrm{H_O}$ and $\mathrm{H_U}$ represent a set of observed elements and a collection of unobserved elements in $\boldsymbol{H}$, respectively. If the condition $|H_O| \gg |H_U|$ holds, then H is the HDI matrix.

Definition 2.2 LFL. Given entity set M and entity set N, given HDI matrix $\mathbf{H}$, given a constant d, given a $|M| \times d$ latent feature matrix $\mathbf{X}$, given a $d \times |N|$ latent feature matrix $\mathbf{Y}$, $\hat{\mathbf{H}}$ is the estimation matrix of $\mathbf{H}$, where each element $\hat{h}_{m,n}$ is an estimate of $h_{m,n}$, d is the latent feature dimension of $\mathbf{X}$ and $\mathbf{Y}$, $d \gg \min\{|M|,|N|\}$. Then, LFL is to train two latent feature matrices $\mathbf{X}$ and $\mathbf{Y}$ by minimizing the estimation errors between $\hat{\mathbf{H}}$ and $\mathbf{H}$ on the known element set H_O, where $\hat{\mathbf{H}}$ can be obtained by the dot product of $\mathbf{X}$ and $\mathbf{Y}$, that is, $\hat{\mathbf{H}} = \mathbf{X}\mathbf{Y}^{\mathbf{T}}$.

2.3 Latent Feature Learning

2.3.1 A Basic LFL Model

Each element in the HDI matrix describes the value of a specific relationship between the two tuples (m, n) of an element in the entity set M and N. Taking the scenario of an e-commerce system as an example. Let M represent the collection of users, N represent the collection of goods, and $|M| \times |N|$ Matrix $\mathbf{H}$ describe the quantified value of users' preference for goods, which is quantified according to the records of users' purchases, comments, and ratings of goods. Obviously, in e-commerce systems that contain tens of millions of users and millions of goods, such as the Alibaba Shopping website and Amazon Shopping website, one user cannot touch all products, and one product cannot be touched by all users. Therefore, under normal circumstances, a user (a commodity) will only contact a very small subset of the commodity (user) collection, that is, the condition is satisfied $|H_O| \gg |H_U|$. Therefore, in the e-commerce system, the user-item preference relationship can be described by the HDI matrix $\mathbf{H}$.

According to Definitions 2.1 and 2.2, LFL is to decompose the original HDI matrix $\mathbf{H}$ into two hidden feature matrices $\mathbf{X}$ and $\mathbf{Y}$ decompose. Figure 2.1 shows the decomposition diagram of LFL. First, initialize two hidden latent feature matrices $\mathbf{X}$ and $\mathbf{Y}$, which can be initialized with non-zero random values; second, train on the set of known elements in the original HDI matrix $\mathbf{H}$, where the whole original HDI matrix $\mathbf{H}$ is not used as the training, but only the known elements are inputted, and

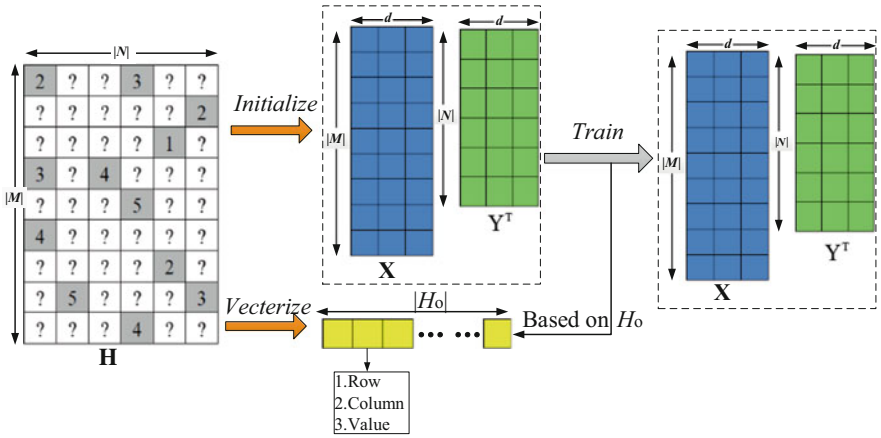


Fig. 2.1 The diagram of LFL

the input data structure is (1. row, 2. column, 3. value). Finally, when the training iteration stop condition is reached, the training is stopped, and the hidden characteristic matrices $\mathbf{X}$ and $\mathbf{Y}$ are output. The original HDI matrix $\mathbf{H}$ is sparse. After hidden feature extraction, two low-dimensional dense hidden feature matrices $\mathbf{X}$ and $\mathbf{Y}$ are obtained. The hidden feature matrix $\mathbf{X}$ corresponds to the rows in the original HDI matrix $\mathbf{H}$, and they have the same number of rows; the hidden feature matrix $\mathbf{Y}$ corresponds to the columns in the original HDI matrix $\mathbf{H}$, which have the same number of columns; in the process of extracting hidden features, $d \gg \min\{|M|,|N|\}$, so the dimensionality reduction of the original HDI matrix $\mathbf{H}$ is realized, that is, the data granulation is realized. In the follow-up big data processing tasks, hidden feature matrices $\mathbf{X}$ and $\mathbf{Y}$ can be used to carry out missing value prediction, cluster analysis, classification analysis, and trend prediction.

In general, the low-dimensional hidden feature space, that is, $d \gg \min\{|M|,|N|\}$, can make $\mathbf{X}$ and $\mathbf{Y}$ represent $\mathbf{H}$ well so that LFL has very low time complexity and space complexity. According to Definition 2.2, the key to constructing hidden feature analysis is how to minimize the estimation errors of $\mathbf{H}$ on the known element set H_O, specifically, how to design the loss function or objective function. In the design of the objective function, Euclidean distance is most used. The objective function designed by Euclidean distance is:

$$\arg\min_{\mathbf{X},\mathbf{Y}} \varepsilon(\mathbf{X},\mathbf{Y}) = \left\| \Omega \odot \left(\mathbf{H} - \widehat{\mathbf{H}} \right) \right\|_F^2 = \left\| \Omega \odot (\mathbf{H} - \mathbf{XY}) \right\|_F^2, \tag{2.1}$$

where, $\varepsilon(\mathbf{X},\mathbf{Y})$ denotes the objective function of LFL with respect to $\mathbf{X}$ and $\mathbf{Y}$. $\|\cdot\|_F$ refers to the Frobenius norm of the matrix, $\odot$ denotes the Hadamard product, and Ω denotes a $|M| \times |N|$ binary index matrix. Ω is calculated as:

$$\Omega_{m,n} = \begin{cases} 1 & \text{if } h_{m,n} \text{ is observed} \\ 0 & \text{otherwise} \end{cases}. \tag{2.2}$$

In the field of machine learning, regularization techniques prevent models from overfitting to known training datasets. To avoid overfitting, Tikhonov regularization can be added to Eq. (2.1) as follows:

$$\arg\min_{M,\mathrm{N}} \varepsilon(M, N) = \Omega\|\mathbf{H} - \mathbf{XY}\|_F^2 + \lambda\left(\|\mathbf{X}\|_F^2 + \|\mathbf{Y}\|_F^2\right), \tag{2.3}$$

where λ is the regularization control factor, and the two terms in parentheses after λ are Tikhonov regularization. According to [24, 25], the introduction of linear deviation into Formula (2.3) can improve the prediction accuracy of LFL under certain circumstances, and then Formula (2.3) is modified as follows:

$$\begin{aligned} \arg\min_{\mathbf{X},\mathbf{Y}} \varepsilon(\mathbf{X}, \mathbf{Y}) &= \Omega\|\mathbf{H} - \mathbf{W} - \mathbf{XY}\|_F^2 \\ &+ \lambda\left(\|\mathbf{X}\|_F^2 + \|\mathbf{Y}\|_F^2 + \|B_n\|_F^2 + \|B_m\|_F^2\right), \end{aligned} \tag{2.4}$$

where $\mathbf{W}$ is a $|M| \times |N|$ linear deviation matrix, each element $w_{m,n}$ in $\mathbf{W}$ refers to the linear deviation with respect to the entity $h_{m,n}$; B_m is a vector with $|M|$ elements, where the Each element b_m refers to the observed linear deviation of entity m; B_n is a vector with $|N|$ elements, where each element bi refers to the observed linear deviation of entity n. $w_{m,n}$ can be calculated by the following formula:

$$w_{u,s} = \mu + b_m + b_n, \tag{2.5}$$

where μ refers to the global draw value of all known entities in $\mathbf{H}$. By minimizing Formula (2.5), the latent feature matrices $\mathbf{X}$ and $\mathbf{Y}$ can be obtained, and then the estimated matrix $\hat{\mathbf{H}}$ of $\mathbf{H}$ can be obtained:

$$\widehat{\mathbf{H}} = W + \mathbf{XY}. \tag{2.6}$$

For LFL, the minimization Formulas (2.5) and (2.6) can be implemented using any optimizer, typically stochastic gradient descent (SGD). In Eqs. (2.5) and (2.6), there are many unknown values in the HDI matrix R, which need to be extended to a single-element form. First, consider the single-element form without the addition of a linear deviation:

$$\begin{aligned}\arg\min_{\mathbf{X},\mathbf{Y}} \varepsilon(\mathbf{X},\mathbf{Y}) &= \sum_{(m,\ n)\in H_{\mathrm{O}}} \left(h_{m,n} - \sum_{k=1}^{d} x_{m,k}y_{k,n}\right)^2 + \lambda \sum_{(m,\ s)\in H_{\mathrm{O}}} \left(\sum_{k=1}^{d} x_{m,k}^2 + \sum_{k=1}^{d} y_{k,n}^2\right) \\ &= \sum_{(m,\ n)\in H_{\mathrm{O}}} \left(\left(h_{m,n} - \sum_{k=1}^{d} x_{m,k}y_{k,n}\right)^2 + \lambda\left(\sum_{k=1}^{d} x_{m,k}^2 + \sum_{k=1}^{d} y_{k,n}^2\right)\right),\end{aligned} \tag{2.7}$$

where $x_{m,k}$ and $y_{k,n}$ refer to the elements in the latent feature matrices $\mathbf{X}$ and $\mathbf{Y}$, respectively. Next, denote the partial loss on the entity $h_{m,n}$ by $\varepsilon_{m,n}$ namely:

$$\varepsilon_{m,n} = \left(h_{m,n} - \sum_{k=1}^{d} x_{m,k}y_{k,n}\right)^2 + \lambda\left(\sum_{k=1}^{d} x_{m,k}^2 + \sum_{k=1}^{d} y_{k,n}^2\right). \tag{2.8}$$

For Formula (2.8), the stochastic gradient descent method is adopted, that is, each single element in the latent feature matrices $\mathbf{X}$ and $\mathbf{Y}$ is moved to the opposite direction of the gradient of Formula (2.8):

$$\text{On } h_{m,n}, \forall k \in \{1,\ 2,\ \ldots,\ d\} : \begin{cases} x_{m,k} = x_{m,k} - \eta\frac{\partial \varepsilon_{m,n}}{\partial x_{m,k}} \\ y_{k,n} = y_{k,n} - \eta\frac{\partial \varepsilon_{m,n}}{\partial y_{k,n}} \end{cases}, \tag{2.9}$$

where η is the learning rate, which controls the pace of gradient descent. Next, Formula (2.9) is extended to obtain updated formulas for each single element $x_{m,k}$ and $y_{k,n}$ in $\mathbf{X}$ and $\mathbf{Y}$:

$$On\ h_{m,n}, for\ k = 1 \sim d : \begin{cases} x_{m,k} \leftarrow x_{m,k} + \eta y_{k,n}\Delta_{m,n} - \lambda\eta x_{m,k} \\ y_{k,n} \leftarrow y_{k,n} + \eta x_{m,k}\Delta_{m,n} - \lambda\eta y_{k,n} \end{cases}, \tag{2.10}$$

where $\Delta_{m,n}$ refers to the prediction error on the entity $h_{m,n}$:

$$\Delta_{m,n} = h_{m,n} - \sum_{k=1}^{d} x_{m,k}y_{k,n}. \tag{2.11}$$

According to Formulas (2.10) and (2.11), the latent feature matrices $\mathbf{X}$ and $\mathbf{Y}$ can be extracted from the original HDI matrix $\mathbf{H}$ without considering the linear deviation, and the prediction of missing values in the original HDI matrix $\mathbf{H}$ can be realized as follows:

$$\widehat{h}_{m,n} = x_{m,\cdot} \cdot y_{\cdot,n}, \tag{2.12}$$

where $\cdot$ denotes the inner product of two vectors.

2.3.2 A Biased LFL Model

Next, considering the LBs to minimize (2.4) in the single-element situation as follows:

$$\arg\min_{\mathbf{X},\mathbf{Y}} \varepsilon(\mathbf{X},\mathbf{Y}) = \sum_{(m,\ n)\in H_{\mathrm{O}}} \left(\left(h_{m,n} - \mu - b_m - b_n - \sum_{k=1}^{d} x_{u,k} y_{k,n} \right)^2 + \lambda \left(\sum_{k=1}^{d} x_{u,k}^2 + \sum_{k=1}^{d} y_{k,n}^2 + {b_m}^2 + {b_n}^2 \right) \right). \tag{2.13}$$

then, the partial loss $\varepsilon_{m,n}$ with respect to the entity $h_{m,s}$ is:

$$\varepsilon_{m,n} = \left(h_{m,n} - \mu - b_m - b_n - \sum_{k=1}^{d} x_{m,k} y_{k,n} \right)^2 + \lambda \left(\sum_{k=1}^{d} x_{m,k}^2 + \sum_{k=1}^{d} y_{k,n}^2 + {b_m}^2 + {b_n}^2 \right), \tag{2.14}$$

where μ, b_m, and b_n can be calculated according to the following formula:

$$\begin{aligned} \mu &= \left(\sum_{(m,\ n)\in H_{\mathrm{O}}} h_{m,n} \right) / |H_{\mathrm{O}}|, \\ b_n &= \left(\sum_{(m,\ n)\in H_{\mathrm{O}}} h_{m,n} - \mu \right) / \left(\theta_1 + \sum_{(m,\ n)\in H_{\mathrm{O}}} 1 \right), \\ b_m &= \left(\sum_{(m,\ n)\in H_{\mathrm{O}}} h_{m,n} - \mu - b_n \right) / \left(\theta_2 + \sum_{(m,\ n)\in H_{\mathrm{O}}} 1 \right), \end{aligned} \tag{2.15}$$

where θ_1 and θ_2 are regularization parameters, which can be obtained by cross-validation. According to Eq. (2.9) and Eq. (2.15), the updated formulas for each single element $x_{m,k}$ and $y_{k,n}$ in $\mathbf{X}$ and $\mathbf{Y}$ can be obtained with the addition of linear deviation:

$$On\ h_{m,n}, for\ k = 1 \sim d : \begin{cases} x_{m,k} \leftarrow x_{m,k} + \eta y_{k,n} (\Delta_{m,n} - \mu - b_m - b_n) - \lambda \eta x_{m,k}, \\ y_{k,n} \leftarrow y_{k,n} + \eta x_{m,k} (\Delta_{m,n} - \mu - b_m - b_n) - \lambda \eta y_{k,n}. \end{cases} \tag{2.16}$$

Similarly, according to Formula (2.12), Formula (2.15), and Formula (2.16), the latent feature matrices **X** and **Y** can be extracted from the original HDI matrix **H** by considering LBs. Then, the prediction of missing values in the original HDI matrix **H** can be realized as follows:

$$\widehat{h}_{m,n} = x_{m,\cdot} \cdot y_{\cdot,n} + \mu + b_m - b_n. \tag{2.17}$$

2.3.3 Algorithms Design

The pseudo code of a basic LFL and a biased LFL models are shown in Tables 2.1 and 2.2, respectively. Their time complexity is as follows.

$$O(N_{mtr} \times |H_{\mathrm{O}}| \times d) \tag{2.18}$$

2.4 Performance Analysis

2.4.1 Evaluation Protocol

In terms of industrial applications, one of the major motivations for analyzing an HDI matrix is to predict its missing entries. To this end, root mean squared error (RMSE) and mean absolute error (MAE) are commonly adopted as the evaluation protocols as follows:

Table 2.1 The pseudo code of a basic LFL model

	Input: H	
	Output: X, Y	Cost
1	Initialize d, λ, η, $N_{mtr} = max\text{-}training\text{-}round$	$\Theta(1)$
2	**while** $t \leq N_{mtr}$ && not converge	$\times N_{\mathrm{mtr}}$
3	**for** each known record $h_{m,n}$ in $\mathbf{H}$//$h_{m,n} \in Ho$	$\times \lvert Ho \rvert$
4	**for** $k = 1$ to d	$\times d$
5	According to Formulas (2.10) and (2.11) to update $x_{m,k}$	$\Theta(1)$
6	According to Formulas (2.10) and (2.11) to update $y_{k,n}$	$\Theta(1)$
7	**end** for	--
8	**end** for	--
9	$t = t + 1$	$\Theta(1)$
10	**end** while	--
11	**Return X, Y**	$\Theta(1)$

Table 2.2 The pseudo code of a biased LFL model

	Input: H	
	Output: X, Y	Cost
1	Initialize d, λ, η, $N_{mtr} = max\text{-}training\text{-}round$	$\Theta(1)$
2	**while** $t \leq N_{mtr}$ && not converge	$\times N_{mtr}$
3	**for** each known record $h_{m,n}$ in **H** // $h_{m,n} \in H_O$	$\times \lvert H_O \rvert$
4	**for** $k = 1$ to d	$\times d$
5	According to Formula (2.11) and (2.16) to update $x_{m,k}$	$\Theta(1)$
6	According to Formula (2.11) and (2.16) to update $y_{k,n}$	$\Theta(1)$
7	**end for**	--
8	**end for**	--
9	$t = t + 1$	$\Theta(1)$
10	**end while**	--
11	**Return X, Y**	$\Theta(1)$

$$RMSE = \sqrt{\left(\sum_{m,\ n \in T} \left(h_{m,n} - \widehat{h}_{m,n}\right)^2\right) / |\Gamma|}, \quad MAE = \left(\sum_{m,\ n \in T} |h_{m,n} - \widehat{h}_{m,n}|_{abs}\right) / |\Gamma|;$$

where Γ denotes the testing set and $|\cdot|_{abs}$ denotes the absolute value of a given number. Note that lower RMSE/MAE denotes a higher prediction accuracy.

2.4.2 Discussion

A specific example will be used to illustrate the case of latent feature extraction from the HDI matrix by LFL, that is, granulation. In this section, the Movielens 1M dataset recognized by the academic community is selected as the experimental data (the dataset can be downloaded from https://grouplens.org/datasets/movielens/), which has been widely adopted by the academic community, especially in the field of recommender systems [20–25]. The dataset includes ratings of 3900 movies by 6040 users. The ratings range from integers in [1, 5], and the known rating items are 1000, 209. The dataset is a typical HDI matrix. The data density is only 4.25%.

In the experiment, Movielens 1M is divided into two parts, 80% as the training dataset and 20% as the test dataset. This process will be repeated five times. To ensure each copy can be a test data set, the average result of 5 results is taken as the result. In terms of parameter settings, the learning rate $\eta = 0.01$, and the regularization factor $\lambda = 0.01$. Note that d is set to 5, 10, 20, 40, 80, 160, and 320 in the experiments, respectively.

Figure 2.2 records the experimental results of the a basic LFL model under different latent feature dimensions d. It can be seen from this that higher hidden feature dimensions achieve higher prediction accuracy and faster convergence speed. However, higher latent feature dimensions require more computing resources in training the model, and Fig. 2.2 shows that when the latent feature dimension

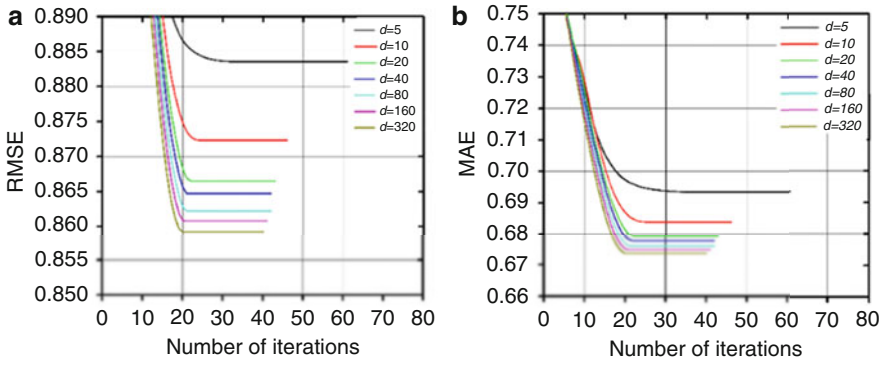


Fig. 2.2 Experimental results of a basic LFL model under different d values

exceeds 20, the prediction accuracy is not improved significantly. Therefore, in the actual big data processing application, the setting of the hidden feature dimension needs to be determined according to the specific analysis task. Generally, d is usually set to be 10–80 dimensions, considering both computational efficiency and prediction accuracy.

LFL can extract latent features from the original HDI matrix. Figure 2.2 shows the distribution of latent features extracted from the Movielens 1M dataset by the basic LFL model with different latent feature dimensions d. It can be seen from Fig. 2.2 that no matter what the hidden feature dimension d is set to, the extracted hidden features obey the normal distribution, and the value range is mainly distributed in $[-2, 2.5]$. When the latent feature dimension d increases gradually, more latent features are extracted, and the distribution is relatively more concentrated, that is, the variance of the normal distribution is smaller.

In the case of considering the LBs, the experimental results of a biased LFL model are similar to that of a basic LFL model, which will not be described in detail here (Fig. 2.3).

Besides, Table 2.3 presents detailed comparisons of prediction accuracy between a basic LFL model and a biased LFL model. A biased LFL mode has a lower MAE/RMSE than a basic LFL model in general.

2.5 Summary

In big data analysis and processing, how to effectively organize and mine massive data is a basic problem to be solved in various applications of big data. This chapter introduces the basis of LFL. First, it introduces the principle of LFL. Then, two LFL models, i.e., a basic LFL model and a bised LFLmodel, are introduced. Finally, some experiments are conducted to evaluate the performance of the two LFL models. The results demonstrate that an LFL model can well extract the latent features from an

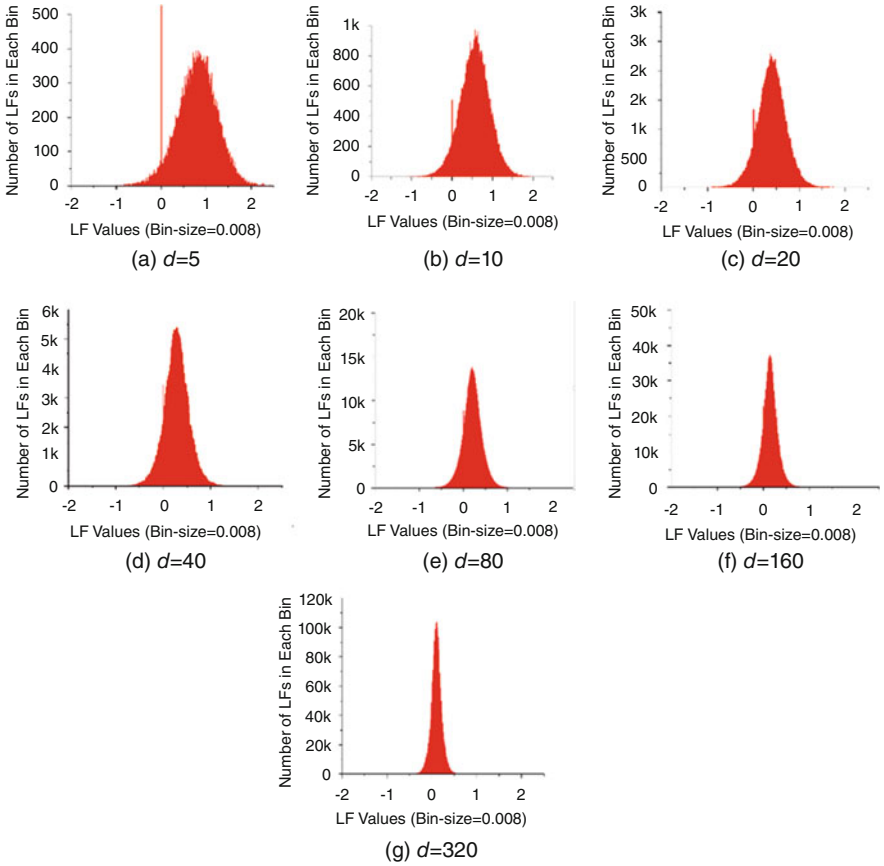


Fig. 2.3 Distribution of latent features extracted by a basic LFL under different d values

Table 2.3 The comparison results on rating prediction accuracy

Dataset	Metric	A basic LFL	A biased LFL	Dataset	Metric	A basic LFL	A biased LFL
D1	MAE	1.2392	1.2328	D6	MAE	0.7664	0.7676
	RMSE	1.8066	1.7809		RMSE	0.9957	0.9982
D2	MAE	0.5537	0.5516	D7	MAE	0.5999	0.6067
	RMSE	0.7139	0.6993		RMSE	0.7819	0.7936
D3	MAE	0.1732	0.1730	D8	MAE	0.5886	0.5955
	RMSE	0.2251	0.2262		RMSE	0.7737	0.7848
D4	MAE	0.3011	0.2938	D9	MAE	0.8037	0.8475
	RMSE	0.5958	0.4821		RMSE	1.0596	1.0968
D5	MAE	0.6447	0.6207	D10	MAE	0.8446	0.8540
	RMSE	0.8961	0.8383		RMSE	0.6634	0.6672

HDI matrix. Besides, the results also show that a biased LFL mode has a lower MAE/RMSE than a basic LFL model in general.

References

1. Xu, X., Wu, D., Shang, M.: A structure-characteristic-aware network embedding model via differential evolution. Expert Syst. Appl. **204**, 117611 (2022)
2. Zhang, P., He, Y., Wu, D.: An ensemble latent factor model for highly accurate web service qos prediction. In: 2021 IEEE International Conference on Big Knowledge (ICBK), pp. 361–368 (2021)
3. You, Z.-H., Zhou, M., Luo, X., Li, S.: Highly efficient framework for predicting interactions between proteins. IEEE Trans. Cybern. **47**(3), 731–743 (2017)
4. He, Y., Wu, D., Beyazit, E., Sun, X., Wu, X.: Supervised data synthesizing and evolving - a framework for real-world traffic crash severity classification. In: IEEE 30th International Conference on Tools with Artificial Intelligence, ICTAI 2018, pp. 163–170 (2018)
5. Luo, X., Liu, Z., Shang, M., Lou, J., Zhou, M.: Highly-accurate community detection via pointwise mutual information-incorporated symmetric non-negative matrix factorization. IEEE Trans. Netw. Sci. Eng. **8**(1), 463–476 (2020)
6. Li, S., You, Z.-H., Guo, H., Luo, X., Zhao, Z.-Q.: Inverse-free extreme learning machine with optimal information updating. IEEE Trans. Cybern. **46**(5), 1229–1241 (2016)
7. Xia, Y., Zhou, M., Luo, X., Zhu, Q., Li, J., Huang, Y.: Stochastic modeling and quality evaluation of infrastructure-as-a-service clouds. IEEE Trans. Autom. Sci. Eng. **12**(1), 162–170 (2015)
8. Xu, X., Pang, G., Wu, D., Shang, M.: Joint hyperbolic and euclidean geometry contrastive graph neural networks. Inf. Sci. **609**, 799–815 (2022)
9. Luo, X., Yuan, Y., Chen, S., Zeng, N., Wang, Z.: Position-transitional particle swarm optimization-incorporated latent factor analysis. IEEE Trans. Knowl. Data Eng. **34**(8), 3958–3970 (2022)
10. Qi, Y., Jin, L., Luo, X., Zhou, M.: Recurrent neural dynamics models for perturbed nonstationary quadratic programs: a control-theoretical perspective. IEEE Trans. Neural Netw. Learn. Syst. **33**(3), 1216–1227 (2022)
11. Lu, H., Jin, L., Luo, X., Liao, B., Guo, D., Xiao, L.: RNN for solving perturbed time-varying underdetermined linear system with double bound limits on residual errors and state variables. IEEE Trans. Industr. Inform. **15**(11), 5931–5942 (2019)
12. Zhang, J.-D., Chow, C.-Y., Xu, J.: Enabling kernel-based attribute-aware matrix factorization for rating prediction. IEEE Trans. Knowl. Data Eng. **29**(4), 798–812 (2017)
13. Gong, M., Jiang, X., Li, H., Tan, K.C.: Multiobjective sparse non-negative matrix factorization. IEEE Trans. Cybern. **49**(99), 1–14 (2018)
14. Li, W., He, Q., Luo, X., Wang, Z.: Assimilating second-order information for building non-negative latent factor analysis-based recommenders. IEEE Trans. Syst. Man Cybern. Syst. **52**(1), 485–497 (2022)
15. Luo, X., Zhou, M.: Effects of extended stochastic gradient descent algorithms on improving latent factor-based recommender systems. IEEE Robot. Autom. Lett. **4**(2), 618–624 (2019)
16. Li, P., Wang, Z., Ren, Z., Bing, L., Lam, W.: Neural rating regression with abstractive tips generation for recommendation. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 345–354 (2017)
17. Luo, X., Zhou, Y., Liu, Z., Zhou, M.: Fast and accurate non-negative latent factor analysis on high-dimensional and sparse matrices in recommender systems. IEEE Trans. Knowl. Data Eng. **1** (2021). https://doi.org/10.1109/TKDE.2021.3125252

18. Castro, J., Lu, J., Zhang, G., Dong, Y., Martínez, L.: Opinion dynamics-based group recommender systems. IEEE Trans. Syst. Man Cybern. Syst. **48**(12), 2394–2406 (2018)
19. Yuan, Y., Luo, X., Shang, M., Wang, Z.: A Kalman-filter-incorporated latent factor analysis model for temporally dynamic sparse data. IEEE Trans. Cybern, 1–14 (2022). https://doi.org/10.1109/TCYB.2022.3185117
20. Li, W., Luo, X., Yuan, H., Zhou, M.: A momentum-accelerated hessian-vector-based latent factor analysis model. Trans. Serv. Comput. (2022). https://doi.org/10.1109/TSC.2022.3177316
21. Wang, Q., Liu, X., Shang, T., Liu, Z., Yang, H., Luo, X.: Multi-constrained embedding for accurate community detection on undirected networks. IEEE Trans. Netw. Sci. Eng. **9**, 3675–3690 (2022). https://doi.org/10.1109/TNSE.2022.3176062
22. Liu, Z., Yuan, G., Luo, X.: Symmetry and nonnegativity-constrained matrix factorization for community detection. IEEE/CAA J. Autom. Sin. **9**, 1691–1693 (2022). https://doi.org/10.1109/JAS.2022.1005794
23. Bi, F., Wu, D.: A proximal alternating-direction-method-of-multipliers-based nonnegative latent factor model. In: 2021 IEEE International Conference on Big Knowledge (ICBK), pp. 353–360 (2021). https://doi.org/10.1109/ICKG52313.2021.00054
24. Yu, R., Liu, Q., Ye, Y., Cheng, M., Chen, E., Ma, J.: Collaborative list-and-pairwise filtering from implicit feedback. IEEE Trans. Knowl. Data Eng. **34**(6), 2667–2680 (2022)
25. Li, H., Diao, X., Cao, J., Zheng, Q.: Collaborative filtering recommendation based on all-weighted matrix factorization and fast optimization. IEEE Access. **6**, 25248–25260 (2018)
26. Chen, J., Wang, R., Wu, D., Luo, X.: A differential evolution-enhanced position-transitional approach to latent factor analysis. In: IEEE Transactions on Emerging Topics in Computational Intelligence, pp. 1–13 (2022)
27. Yu, Z., Wu, D., He, Y.: A robust latent factor analysis model for incomplete data recovery in wireless sensor networks. In: 2022 IEEE International Conference on Edge Computing and Communications (EDGE), pp. 178–183 (2022)
28. Wu, D., He, Y., Luo, X., Zhou, M.: A latent factor analysis-based approach to online sparse streaming feature selection. IEEE Trans. Syst. Man Cybern. Syst. **52**, 6744–6758 (2021)
29. He, X., Lin, X., Wu, D., Wang, J.: An ensemble classification framework based on latent factor analysis. In: 2020 IEEE International Conference on Human-Machine Systems (ICHMS), pp. 1–6. https://doi.org/10.1109/ICHMS49158.2020.9209463
30. Luo, X., Zhou, M., Li, S., You, Z., Xia, Y., Zhu, Q., Leung, H.: An efficient second-order approach to factorize sparse matrices in recommender systems. IEEE Trans. Ind. Inform. **11**(4), 946–956 (2015)
31. Luo, X., Liu, Z., Jin, L., Zhou, Y., Zhou, M.: Symmetric nonnegative matrix factorization-based community detection models and their convergence analysis. IEEE Trans. Neural Netw. Learn. Syst. **33**, 1203 (2022)
32. Shi, X., Luo, X., Shang, M., Gu, L.: Long-term performance of collaborative filtering based recommenders in temporally evolving systems. Neurocomputing. **267**, 635–643 (2017)

Chapter 3
Robust Latent Feature Learning based on Smooth L_1-norm

3.1 Overview

In this era of information explosion, big data for various industrial applications are surrounding people [1–5], such as recommendation systems (RSs), social networks, wireless sensor networks, and intelligent transportation. In these applications, the relationship between the two types of entities is usually represented by a matrix. For example, it is common to see a user-item rating matrix in RSs [6–9], where each row represents a specific user, each column represents a specific item, and each entry represents the user's preference for an item.

In big data applications such as Amazon, because the relationships between many entities are not able to be fully observed in practice, the matrix from these applications is usually high dimensional and incomplete (HDI) [7, 10, 11]. There, how accurately extracting useful information from an HDI matrix has become a hot and difficult problem in industrial applications.

So far, to solve this problem, various researchers have proposed various methods and models [6–9]. Among them, latent feature learning (LFL) is one of the most popular approaches. How to design the LFL's loss function is very important, in which the loss function refers to the sum of errors between the ground truth data and the predicted data.

Commonly, L_2 norm-oriented *Loss* is adopted by most LFL models [12–18], while L_1 norm-oriented *Loss* is rarely used in an LFL model [19]. Figure 3.1 illustrates the difference between the norm steering losses of L_1 and L_2-norm-oriented *Losses* [19–21], where *Error* represents the error between the predicted results and the real data. Figure 3.1 shows that: (a) the L_1 norm is less sensitive to *Errors*, making the resultant model more robust [19, 21, 22], and (b) when the absolute value of *Error* is small (less than 1), the L_2 norm is smoother than the L_1 norm, thus enhancing the stability of a resulting model [23].

Unfortunately, an HDI matrix from real applications is usually mixed with outlier data. For example, the HDI matrix from RSs usually contains many outlier ratings

D. Wu, *Robust Latent Feature Learning for Incomplete Big Data*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-19-8140-1_3

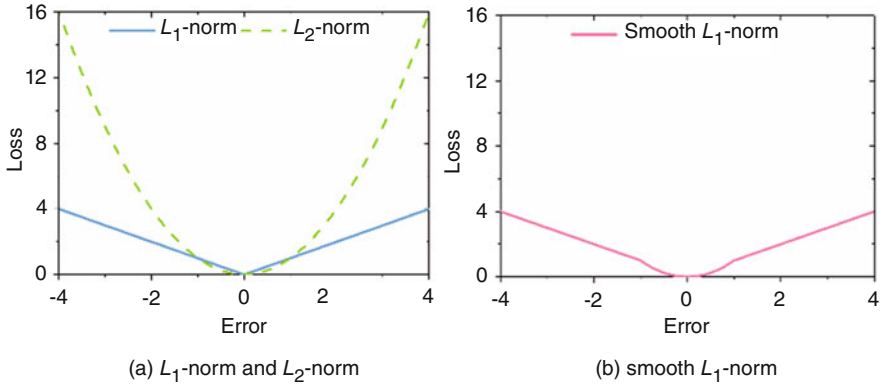


Fig. 3.1 The relationship between *Loss* and *Error* with different norms to solve a regression problem. This figure comes from [51]

[24, 25] due to careless/malicious users. Hence, it is required to use L_1 norm to improve an LFL model's robustness. Therefore, the best choice for building an LFL model is not a single loss guided by the L_1 and L_2 norms. Based on this observation, this chapter introduces a smooth L_1-norm-oriented latent feature (SL-LF) model. Its main idea is to use the smooth L_1-norm to form the *Loss* to ensure its strong robustness and high accuracy in predicting the missing data of the HDI matrix.

3.2 Related Work

It has been proven that an LFL model is one of the most successful and popular methods to predict missing data of an HDI matrix. So far, many researchers have proposed a variety of methods, including probabilistic [43], non-negativity-constrained [14], multiplicative update-based [44], randomized [17], posterior-neighborhood-regularized [16], data characteristic-aware [35], nonnegative-based [37], graph regularized [18], neighborhood-and-location integrated [6], confidence-driven [45], deep-latent-factor based [46], and triple factorization-based [47] ones. They all use L_2 norm-oriented *Loss*, making they be very sensitive to outlier data [20].

To reduce the sensitivity of an LFL model to outlier data, Zhu et al. suggested to adopt the L_1-norm-oriented *Loss* [19]. Different from these methods, SL-LF model uses a smooth L_1-norm-oriented *Loss*, so that its solution space is smoother and has less multimodal than that of an LFL model with L_1-norm-oriented *Loss*. At the same time, its robustness is also higher than that of an LFL model with L_2-norm-oriented *Loss*.

Recently, the representation of an HDI matrix based on a deep neural network (DNN) has attracted much attention [48–50]. According to [34], researchers have proposed various models, including autoencoder-based [36] and hybrid

autoencoder-based models [38]. However, the DNN-based model has the limitation of high computational cost because of its learning strategies. In contrast, SL-LF only trains the known data of the HDI matrix, thus achieving high computational efficiency. It has some possible applications in presentation learning, such as automatic vehicles, community detection, and medical image analysis.

3.3 A Smooth L_1-Norm Based Latent Feature Model

3.3.1 Objective Formulation

Because L_2-norm is sensitive to outlier data [20], an LFL model with L_2-norm-oriented *Loss* lacks robustness. On the other hand, the LFL model with L_1-norm-oriented *Loss* function cannot guarantee a high prediction accuracy. As shown in Fig. 3.1(b), the smooth L_1-norm has the advantages of both robustness and smoothness. First, the smooth L_1-norm of the matrix is defined.

Definition 3.1 (Smooth L_1-norm of a matrix). Given a matrix $\mathbf{H}$ with M rows and N columns, the symbol $\|\mathbf{H}\|_{SL1}$ denotes the smooth L_1-norm of $\mathbf{H}$.

$$\|H\|_{SL_1} = \sum_{m=1}^{M} \sum_{n=1}^{N} v, \ v = \begin{cases} \sqrt{h_{m,n}^2} & if \ |h_{mn}| \le 1 \\ |h_{mn}| & if \ |h_{mn}| > 1 \end{cases} \tag{3.1}$$

where $h_{m,n}$ denotes $\mathbf{H}$'s element at m-th row and n-th column, $m \in \{1, 2, \ldots, M\}$, $n \in \{1, 2, \ldots, N\}$.Then, it designs the following objective function as follows:

$$\arg\min_{X,Y} \varepsilon(X, Y) = \left\| \Omega \bigodot (H - XY^T) \right\|_{SL_1}^2 + \lambda \left(\|X\|_{L_2}^2 + \|Y\|_{L_2}^2 \right) \tag{3.2}$$

where $\|\cdot\|_{SL1}$ denotes the smooth L_1-norm of a matrix. As $\mathbf{H}$ contains numerous unknown data, the following data density-oriented forms should extend (3.2) to improve storage and computing efficiency [12, 14]:

$$\arg\min_{X,Y} \varepsilon(X, Y) = \begin{cases} \sum\limits_{(m,\ n) \in H_o} (\Delta_{m,n})^2 + \lambda \sum\limits_{(m,\ n) \in H_o} \left(\sum\limits_{k=1}^{d} x_{m,k}^2 + \sum\limits_{k=1}^{d} y_{n,k}^2 \right) & if \ |\Delta_{m,n}| \le 1 \\ \sum\limits_{(m,\ n) \in H_o} |\Delta_{m,n}| + \lambda \sum\limits_{(m,\ n) \in H_o} \left(\sum\limits_{k=1}^{d} x_{m,k}^2 + \sum\limits_{k=1}^{d} y_{n,k}^2 \right) & if |\Delta_{m,n}| > 1 \end{cases}, \tag{3.3}$$

where $x_{m,k}$ denotes the specific entity at the m-th row and k-th column in $\mathbf{X}$, $y_{n,k}$ denotes the specific entity at the n-th row and k-th column in $\mathbf{Y}$, and $\Delta_{m,n}$ is the prediction error between $h_{m,n}$ and $\hat{h}_{m,n}$ calculated as follows:

$$\Delta_{m,n} = h_{m,n} - \widehat{h}_{m,n} = h_{m,n} - \sum_{k=1}^{d} x_{m,k} y_{n,k}. \tag{3.4}$$

3.3.2 Model Optimization

With such an objective function of (3.3), an SL-LF model is checked whether it can achieve a highly robust and accurate prediction for the missing data of an HDI matrix.

As analyzed in [12, 14], a stochastic gradient descent (SGD) algorithm is adopted to minimize (3.3) by training the desired $\mathbf{X}$ and $\mathbf{Y}$. First, considering the instantaneous loss of (3.3) on a single entity $h_{u,i}$:

$$\varepsilon_{m,n} = \begin{cases} (\Delta_{m,n})^2 + \lambda\left(\sum_{k=1}^{d} x_{m,k}^2 + \sum_{k=1}^{d} y_{n,k}^2\right) & if\ |\Delta_{m,n}| \le 1 \\ |\Delta_{m,n}| + \lambda\left(\sum_{k=1}^{d} x_{m,k}^2 + \sum_{k=1}^{d} y_{n,k}^2\right) & if |\Delta_{m,n}| > 1 \end{cases}. \tag{3.5}$$

To solve (3.5) with SGD, it needs to respectively move each single LF $x_{m,k}$ and $y_{n,k}$ along the opposite of the stochastic gradient of (3.5) as follows:

$$\begin{cases} x_{m,k} = x_{m,k} - \eta \dfrac{\partial \varepsilon_{m,n}}{\partial x_{m,k}} \\ y_{n,k} = y_{n,k} - \eta \dfrac{\partial \varepsilon_{m,n}}{\partial y_{n,k}} \end{cases}, \forall k \in \{1, 2, \ldots, d\} \tag{3.6}$$

where η is the learning rate. Since (3.5) has the absolute deviation term of $|\Delta_{m,n}|$, there are a total of three situations for (3.5) in computing (3.6) as follows,

$$\varepsilon_{m,n} = \begin{cases} (\Delta_{m,n})^2 + \lambda\left(\sum_{k=1}^{d} x_{m,k}^2 + \sum_{k=1}^{d} y_{n,k}^2\right) & if |\Delta_{m,n}| \le 1 \\ \Delta_{m,n} + \lambda\left(\sum_{k=1}^{d} x_{m,k}^2 + \sum_{k=1}^{d} y_{n,k}^2\right) & if \Delta_{m,n} > 1 \\ -\Delta_{m,n} + \lambda\left(\sum_{k=1}^{d} x_{m,k}^2 + \sum_{k=1}^{d} y_{n,k}^2\right) & if \Delta_{m,n} < -1 \end{cases}. \tag{3.7}$$

Then, by incorporating (3.7) into (3.6), it can obtain the training rules for each single latent feature $x_{m,k}$ and $y_{n,k}$ respectively on a single entity $h_{m,n}$ as follows:

$$\text{For } h_{m,n}, \forall k \in \{1, 2, \ldots, d\}:$$
$$\begin{cases} \begin{cases} x_{m,k} = x_{m,k} + \eta y_{n,k} \Delta_{m,n} - \eta \lambda x_{m,k} \\ y_{n,k} = y_{n,k} + \eta x_{m,k} \Delta_{m,n} - \eta \lambda y_{n,k} \end{cases} & if\ |\Delta_{m,n}| \le 1 \\ \begin{cases} x_{m,k} = x_{m,k} + \eta y_{n,k} - \eta \lambda x_{m,k} \\ y_{n,k} = y_{n,k} + \eta x_{m,k} - \eta \lambda y_{n,k} \end{cases} & if\ \Delta_{m,n} > 1 \\ \begin{cases} x_{u,k} = x_{u,k} - \eta y_{n,k} - \eta \lambda x_{u,k} \\ y_{n,k} = y_{n,k} - \eta x_{m,k} - \eta \lambda y_{n,k} \end{cases} & if\ \Delta_{m,n} < -1 \end{cases} . \tag{3.8}$$

After all the known entities in H_o are employed to train by (3.8), the desired $\mathbf{X}$ and $\mathbf{Y}$ are obtained. Finally, $\mathbf{H}$'s rank-f approximation $\widehat{\mathbf{H}}$, which can predict the missing data of $\mathbf{H}$, is obtained as $\widehat{\mathbf{H}} = \mathbf{X}\mathbf{Y}^{\mathrm{T}}$.

3.3.3 *Incorporating Linear Biases into SL-LF*

According to [25–28], an LFL model can incorporate linear biases to improve its prediction accuracy.Note that SL-LF is also an LFL-based model, it can be extended with linear biases. According to [25–28], linear biases usually include the global average and the observed deviations. Then, an SL-LF model with linear biases approximates $\widehat{h}_{m,n}$ as follows.

$$\widehat{h}_{m,n} = \mu + b_m + b_n + \sum_{k=1}^{d} x_{m,k} y_{n,k}, \tag{3.9}$$

where μ denotes the global average value in H_o, b_m denotes the observed deviations on the m-th row of $\mathbf{H}$ matrix, and b_n denotes the observed deviations on the n-th row of $\mathbf{H}$ matrix, respectively. Given an HDI matrix $\mathbf{H}$, computing μ as follows.

$$\mu = \frac{\sum\limits_{(m,\ n) \in H_o} h_{m,n}}{|H_o|}. \tag{3.10}$$

After that, b_m and b_n can be estimated as follows [27, 28]:

$$b_n = \frac{\sum\limits_{(m,\ n)\in H(n)} (h_{m,n} - \mu)}{\theta_1 + |H(n)|}, \quad b_m = \frac{\sum\limits_{(m,\ n)\in H(m)} (h_{m,n} - \mu - b_n)}{\theta_2 + |H(m)|}, \tag{3.11}$$

where $H(n)$ denotes the known entities set in the n-th column of $\mathbf{H}$, $H(u)$ denotes the known entities set in the m-th row of $\mathbf{H}$, θ_1 and θ_2 denote the threshold constant [28]. Then, (3.4) incorporates linear biases into SL-LF, and is changed into:

$$\Delta_{m,n} = h_{m,n} - \widehat{h}_{u,i} = \widehat{h}_{m,n} - \mu - b_m - b_n - \sum_{k=1}^{d} x_{m,k} y_{n,k}. \tag{3.12}$$

Finally, by combining (3.12) into (3.8), training rules for SL-LF with linear biases can be obtained on a single entity $h_{m,n}$.

Notably, SL-LF model has two versions, i.e., without and with linear biases. SL-LF without and with linear biases are termed as SL-LF$_{\bar{b}}$ and SL-LF$_b$, respectively, in the next sections.

3.4 Performance Analysis

3.4.1 General Settings

Datasets For each dataset, 80% of known data are used as a training dataset and the remaining 20% as a testing dataset. Five-fold cross-validations are adopted. Table 3.1 summarizes the adopted datasets.

Baselines SL-LF is compared with five related state-of-the-art models. Table 3.2 gives a brief introduction to these models. To make a fair comparison, d is set as 20 for all the LFL-based models and the SL-LF model. To evaluate prediction accuracy, mean absolute error (MAE) and root mean squared error (RMSE) are computed.

Table 3.1 Properties of all the datasets

No.	Name	|M|	|N|	|H$_o$|	Density (%)
D1	Dating [30]	135,359	168,791	17,359,346	0.08
D2	Douban [12, 29]	129,490	58,541	16,830,839	0.22
D3	Eachmovie [31]	72,916	1628	2,811,718	2.37
D4	Epinion [29]	755,760	120,492	13,668,321	0.02
D5	Flixter [32]	147,612	48,794	8,196,077	0.11
D6	Jester [32]	24,983	100	1,186,324	47.49
D7	MovieLens_10M [33]	71,567	65,133	10,000,054	0.21
D8	MovieLens_20M [33]	138,493	26,744	20,000,263	0.54

This table comes from [51]

Table 3.2 Descriptions of all the comparison models

Model	Description
BLF	The basic LFL model was introduced in Chap. 2.
NLF	The regularized non-negative LFL model was proposed in 2016 [14]. It improves the L_2-LF model by introducing the non-negative constraint into the objective function design.
FNLF	A fast non-negative LFL model based on generalized momentum was proposed in 2018 [7]. It improves the NLF model.
AutoRec	A DNN-based model was proposed in 2015 [36]. It is a representative model in DNN-based RSs.
DCCR	A DNN-based model was proposed in 2019 [38]. It improves AutoRec by using two different neural networks.
SL-$LF_{\bar{b}}$	The SL-LF model without linear biases.
SL-LF_b	The SL-LF model with linear biases.

This table comes from [51]

3.4.2 Performance Comparison

3.4.2.1 Comparison of Prediction Accuracy

Table 3.3 provides detailed comparison results. These comparative results are statistically analyzed. First is the win/loss counts of SL-$LF_{\bar{b}}$/SL-LF_b versus other models. Second is the Friedman test [39] on these comparison results. These results show that (a) SL-$LF_{\bar{b}}$ and SL-LF_b achieve lower RMSE/MAE than the other models in most testing cases, and (b) SL-LF_b has the lowest F-rank value among all the models. Hence, SL-LF_b has the highest prediction accuracy.

Next, conducting the Wilcoxon signed-ranks test [40, 41] on the comparison results of Table 3.3. Wilcoxon signed-ranks test has three indicators—$R+$, $R-$, and p-value. The larger $R+$ value denotes higher accuracy and the p-value denotes the significance level. Table 3.4 represents the results, which show that SL-LF_b has a significantly higher prediction accuracy than all the comparison models with a significance level of 0.05 except for SL-$LF_{\bar{b}}$. However, SL-LF_b has a much larger $R+$ value than SL-$LF_{\bar{b}}$, which shows that linear biases can improve the prediction accuracy of SL-LF model.

3.4.2.2 Comparison of Computational Efficiency

Their CPU running times is measured to compare the computational efficiency. Figure 3.2 shows the results, which show that:

- DNN-based models (DCCR and AutoRec) spend much more time than the other models because of the DNN-based learning strategy that is time-consuming [42].
- SL-LF costs slightly more time than BLF. The reason is that SL-LF has the additional computational procedures of discrimination while BLF does not.

Table 3.3 The comparison results on prediction accuracy, including win/loss counts statistic and Friedman test, where ●and ✪ respectively indicate that SL-$LF_{\bar{b}}$, and SL-LF_b have a higher prediction accuracy than comparison models

Dataset	Metric	BLF	NLF	FNLF	AutoRec	DCCR	SL-$LF_{\bar{b}}$	SL-LF_b
D1	MAE	1.2392●✪	1.2617●✪	1.2588●✪	1.2610●✪	1.2574●✪	1.1702	1.1686
	RMSE	1.8066✪	1.8245✪	1.8215✪	1.8027✪	1.8013✪	1.8275	1.7829
D2	MAE	0.5537●✪	0.5590●✪	0.5592●✪	0.5606●✪	0.5581●✪	0.5516	0.5458
	RMSE	0.7139●✪	0.7150●✪	0.7139●✪	0.7080✪	0.7074✪	0.7094	0.6957
D3	MAE	0.1732●✪	0.1767●✪	0.1763●✪	0.1784●✪	0.1775●✪	0.1731	0.1729
	RMSE	0.2251●	0.2264●✪	0.2259●	0.2305●✪	0.2289●✪	0.2242	0.2260
D4	MAE	0.3011●✪	0.3047●✪	0.3036●✪	0.3014●✪	0.3036●✪	0.2967	0.2766
	RMSE	0.5958✪	0.5994●✪	0.5977✪	0.5946✪	0.5952✪	0.5992	0.4812
D5	MAE	0.6447●✪	0.6550●✪	0.6520●✪	0.6295✪	0.6308✪	0.6348	0.6084
	RMSE	0.8961●✪	0.9056●✪	0.9038●✪	0.8682✪	0.8792✪	0.8949	0.8324
D6	MAE	0.7664●✪	0.7769●✪	0.7778●✪	0.7905●✪	0.7883●✪	0.7552	0.7573
	RMSE	0.9957✪	1.0049●✪	1.0003●✪	1.0078●✪	1.0042●✪	0.9988	0.9936
D7	MAE	0.5999●✪	0.6080●✪	0.6068●✪	0.6048●✪	0.6002●✪	0.5950	0.5980
	RMSE	0.7819●	0.7893●✪	0.7881●	0.7865●	0.7847●	0.7806	0.7887
D8	MAE	0.5886●✪	0.5977●✪	0.5961●✪	0.5947●✪	0.5902●✪	0.5841	0.5857
	RMSE	0.7737●	0.7819●✪	0.7798●✪	0.7802●✪	0.7789●	0.7730	0.7790
Statistical Analysis	● win/loss	13/3	15/1	14/2	11/5	11/5	–	–
	✪ win/loss	13/3	16/0	14/2	15/1	14/2	–	–
	F-rank*	3.281	6.313	5.125	4.813	3.969	2.625	1.875

This table comes from [51]

Table 3.4 Statistical results on Table 3.3 by conducting the Wilcoxon signed-ranks test

Comparison	R+	R−	p-value
SL-LF$_b$ vs. BLF	121	15	0.0033
SL-LF$_b$ vs. NLF	136	0	0.0002
SL-LF$_b$ vs. FNL	133	3	0.0004
SL-LF$_b$ vs. AutoRec	134	2	0.0004
SL-LF$_b$ vs. DCCR	131	5	0.0006
SL-LF$_b$ vs. SL-LF$_{\bar{b}}$	100	37	0.0544

This table comes from [51]

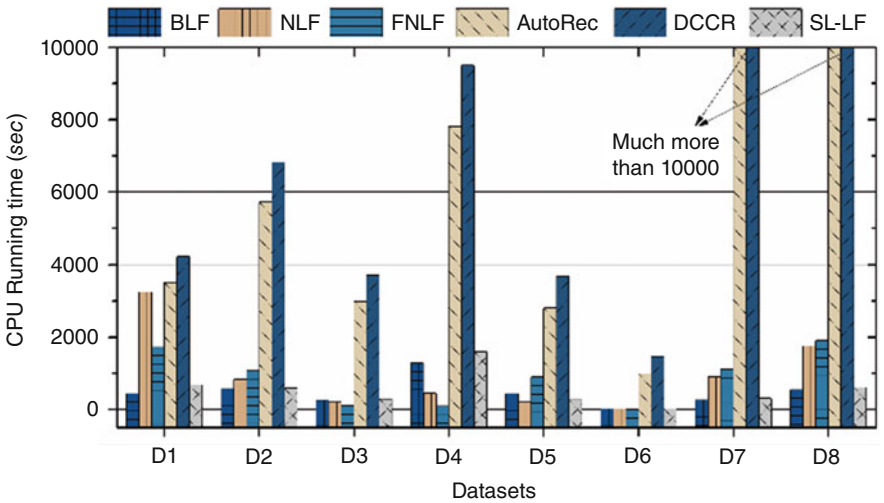


Fig. 3.2 The comparison CPU running time of involved models on D1–D8. This figure comes from [51]

- SL-LF costs less or more CPU running time than NLF and FNLF on the different datasets.

Hence, these results validate that SL-LF's computational efficiency is higher than those of DNN-based models and comparable to those of other LF-based models.

3.4.3 Outlier Data Sensitivity Tests

In this section, the differences between the SL-LF model and the basic LF (BLF) model are compared when adding outlier data to the dataset. The specific method of adding outlier data is: (a) randomly selecting an unknown entity between two known entities as the outlier entity for the input HDI matrix **H**, (b) assigning a value (maximum or minimum known value) to the outlier entity, (c) increase the percentage of outlier entities in known entities from 0% to 100% at 10% intervals, and

(d) only the training set add outlier entities. To illustrate this method, an example is given in Fig. 3.3.

Figure 3.4 records the experimental results on D8. It can be found that with the increase in the percentage of outlier data, both SL-LF$_{\bar{b}}$ and SL-LF$_b$ become much more robust than BLF. Therefore, an SL-LF model is robust to the outlier data.

3.4.4 The Impact of Hyper-Parameter

This set of experiments increases λ from 0.01 to 0.1 and η from 0.0001 to 0.01 by performing a grid-based search. Figures 3.5 and 3.6 show the results on D8. It can be found that:

- Both λ and η have a significant impact on the prediction accuracy of both SL-LF$_{\bar{b}}$ and SL-LF$_b$. As λ and η increase, MAE and RMSE decrease at first and then increase in general.

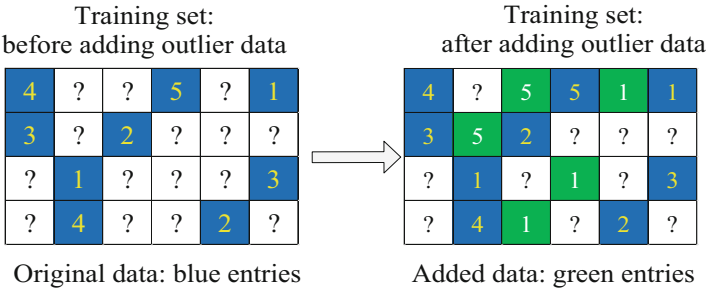


Fig. 3.3 An example of adding outlier data. This figure comes from [51]

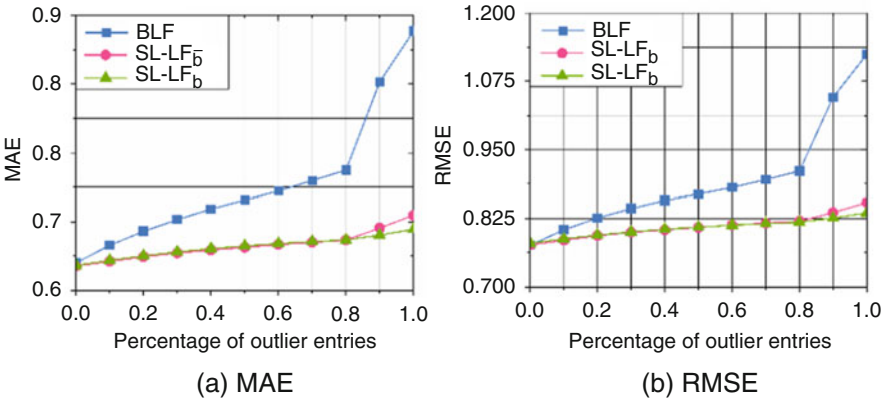


Fig. 3.4 The outlier data sensitivity tests results of BLF, SL-LF$_{\bar{b}}$, and SL-LF$_b$ on D8, where $\lambda = 0.01$, $\eta = 0.0001$, and $d = 20$. This figure comes from [51]

- λ and η have different situations in searching their optimal values on the tested datasets.

3.5 Summary

In this chapter, a Smooth L_1-norm-oriented Latent Feature (SL-LF) model is introduced to predict the missing data of an HDI matrix robustly and accurately. A large number of experiments on eight HDI matrices from industrial applications are conducted to evaluate the SL-LF model. The experimental results verify that: (a) SL-LF is robust to the outlier data, (b) SL-LF significantly outperforms state-of-the-art models in terms of prediction accuracy for the missing data of an HDI

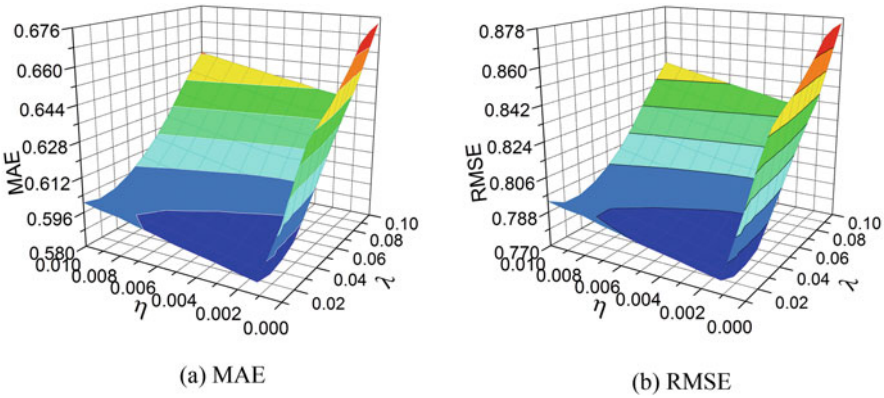


Fig. 3.5 The experimental results of SL-LF$_{\text{b}^-}$ for λ and η on D8, where $d = 20$. This figure comes from [51]

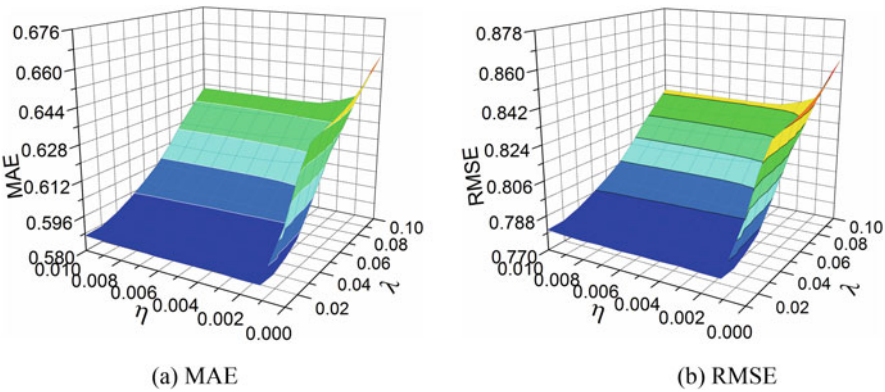


Fig. 3.6 The experimental results of SL-LF$_{\text{b}}$ for λ and η on D8, where $d = 20$. This figure comes from [51]

matrix, and (c) SL-LF's computational efficiency is much higher than those of DNN-based models and comparable to those of most efficient LFL models. Notably, some researchers incorporate non-negative constraints into the LFL model to improve its performance. Similarly, SL-LF could be enhanced by considering non-negative constraints in the future.

References

1. Yuan, Y., Luo, X., Shang, M., Wang, Z.: A Kalman-filter-incorporated latent factor analysis model for temporally dynamic sparse data. IEEE Trans. Cybern., 1–14 (2022). https://doi.org/10.1109/TCYB.2022.3185117
2. Li, W., Luo, X., Yuan, H., Zhou, M.: A momentum-accelerated Hessian-vector-based latent factor analysis model. IEEE Trans. Serv. Comput. (2022). https://doi.org/10.1109/TSC.2022.3177316
3. Liu, Z., Yuan, G., Luo, X.: Symmetry and nonnegativity-constrained matrix factorization for community detection. IEEE/CAA J. Autom. Sin. **9**, 1691–1693 (2022). https://doi.org/10.1109/JAS.2022.1005794
4. Luo, X., Wu, H., Li, Z.: NeuLFT: a novel approach to nonlinear canonical polyadic decomposition on high-dimensional incomplete tensors. IEEE Trans. Knowl. Data Eng., 1 (2022); Y. Ma, Z. Wang, H. Yang, and L. Yang, "Artificial intelligence applications in the development of autonomous vehicles: a survey,"*IEEE/CAA J. Autom Sin,* 7(2), 315–329 (2020)
5. Luo, X., Wu, H., Yuan, H., Zhou, M.: Temporal pattern-aware QoS prediction via biased non-negative latent factorization of tensors. IEEE Trans. Cybern. **50**(5), 1798–1809 (2019)
6. Luo, X., Liu, Z., Li, S., Shang, M., Wang, Z.: A fast non-negative latent factor model based on generalized momentum method. IEEE Trans. Syst. Man Cybern. Syst. **51**(1), 610–620 (2018)
7. Wu, D., Luo, X., He, Y., Zhou, M.: A prediction-sampling-based multilayer-structured latent factor model for accurate representation to high-dimensional and sparse data. In: IEEE Transactions on Neural Networks and Learning Systems, pp. 1–14 (2022)
8. Wu, D., Zhang, P., He, Y., Luo, X.: A double-space and double-norm ensembled latent factor model for highly accurate web service qos prediction. IEEE Trans. Serv. Comput., 1 (2022). https://doi.org/10.1109/TSC.2022.3178543
9. Luo, X., Yuan, Y., Wu, D.: Adaptive regularization-incorporated latent factor analysis. In: 2020 IEEE International Conference on Knowledge Graph (ICKG), pp. 481–488 (2020)
10. He, X., Tang, J., Du, X., Hong, R., Ren, T., Chua, T.: Fast matrix factorization with nonuniform weights on missing data. In: IEEE Transactions on Neural Networks and Learning Systems, pp. 1–14 (2019)
11. Wu, D., He, Y., Luo, X., Shang, M., Wu, X.: Online feature selection with capricious streaming features: A general framework. In: 2019 IEEE International Conference on Big Data (Big Data), pp. 683–688 (2019)
12. Luo, X., Zhou, M., Xia, Y., Zhu, Q.: An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. IEEE Trans. Ind. Inform. **10**(2), 1273–1284 (2014)
13. Wang, Q., Liu, X., Shang, T., Liu, Z., Yang, H., Luo, X.: Multi-constrained embedding for accurate community detection on undirected networks. IEEE Trans. Netw. Sci. Eng. **9**, 3675–3690 (2022). https://doi.org/10.1109/TNSE.2022.3176062
14. Luo, X., Zhou, M., Li, S., You, Z., Xia, Y., Zhu, Q.: A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method. IEEE Trans. Neural Netw. Learn. Syst. **27**(3), 579–592 (2016)

15. Wu, D., Shang, M., Luo, X., Wang, Z.: An L_1-and-L_2-norm-oriented latent factor model for recommender systems. In: IEEE Transactions on Neural Networks and Learning Systems, pp. 1–14 (2021)
16. Wu, D., He, Q., Luo, X., Shang, M., He, Y., Wang, G.: A posterior-neighborhood-regularized latent factor model for highly accurate web service qos prediction. IEEE Trans. Serv. Comput. **15**(2), 793–805 (2022)
17. Shang, M., Luo, X., Liu, Z., Chen, J., Yuan, Y., Zhou, M.: Randomized latent factor model for high-dimensional and sparse matrices from industrial applications. IEEE/CAA J. Autom. Sin. **6**(1), 131–141 (2019)
18. Leng, C., Zhang, H., Cai, G., Cheng, I., Basu, A.: Graph regularized Lp smooth non-negative matrix factorization for data representation. IEEE/CAA J. Autom. Sin. **6**(2), 584–595 (2019)
19. Zhu, X., Jing, X.-Y., Wu, D., He, Z., Cao, J., Yue, D., Wang, L.: Similarity-maintaining privacy preservation and location-aware low-rank matrix factorization for QoS prediction based web service recommendation. IEEE Trans. Serv. Comput. **14**(3), 889–902 (2021)
20. Wang, L., Gordon, M.D., Zhu, J.: Regularized least absolute deviations regression and an efficient algorithm for parameter tuning. In: Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2006), pp. 690–700 (2006)
21. Ma, W., Li, N., Li, Y., Duan, J., Chen, B.: Sparse normalized least mean absolute deviation algorithm based on unbiasedness criterion for system identification with noisy input. IEEE Access. **6**, 14379–14388 (2018)
22. Ke, Q., Kanade, T.: Robust Subspace Computation Using *L1* Norm: School of Computer Science. Carnegie Mellon University, Pittsburgh, PA (2003)
23. Koenker, R., Hallock, K.F.: Quantile regression. J. Econ. Perspect. **15**(4), 143–156 (2001)
24. Wu, C., Qiu, W., Zheng, Z., Wang, X., Yang, X.: Qos prediction of web services based on two-phase k-means clustering. In: Proceedings of 2015 IEEE International Conference on Web Services, pp. 161–168 (2015)
25. Lakshminarayanan, B., Bouchard, G., Archambeau, C.: Robust Bayesian matrix factorisation. In: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, pp. 425–433 (2011)
26. Wu, D., He, Y., Luo, X., Shang, M., Wu, X.: Online feature selection with capricious streaming features: a general framework. In: Proceedings of the 2019 IEEE International Conference on Big Data, pp. 683–688 (2019)
27. Yuan, Y., Luo, X., Shang, M.-S.: Effects of preprocessing and training biases in latent factor models for recommender systems. Neurocomputing. **275**, 2019–2030 (2018)
28. Luo, X., Wu, H., Li, Z.: NeuLFT: a novel approach to nonlinear canonical polyadic decomposition on high-dimensional incomplete tensors. IEEE Trans. Knowl. Data Eng., 1 (2022). https://doi.org/10.1109/TKDE.2022.3176466
29. Massa, P., Avesani, P.: Trust-aware recommender systems. In: Proceedings of the 2007 ACM Conference on Recommender Systems, pp. 17–24. ACM (2007)
30. Brozovsky, L., Petricek, V.: Recommender system for online dating service. arXiv preprint cs/0703042 (2007)
31. Shi, Y., Larson, M., Hanjalic, A.: Collaborative filtering beyond the user-item matrix: a survey of the state of the art and future challenges. ACM Comput. Surv. **47**(1), 1–45 (2014)
32. Goldberg, K., Roeder, T., Gupta, D., Perkins, C.: Eigentaste: a constant time collaborative filtering algorithm. Inf. Retr. **4**(2), 133–151 (2001)
33. Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., Riedl, J.: GroupLens: applying collaborative filtering to usenet news. Commun. ACM. **40**(3), 77–87 (1997)
34. Zhang, S., Yao, L., Sun, A., Tay, Y.: Deep learning based recommender system: a survey and new perspectives. ACM Comput. Surv. **52**(1), 38 (2019)
35. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Wu, X.: A data-characteristic-aware latent factor model for web service QoS prediction. IEEE Trans. Knowl. Data Eng. **34**(6), 2525–2538 (2022)

36. Sedhain, S., Menon, A.K., Sanner, S., Xie, L.: AutoRec: autoencoders meet collaborative filtering. In: Proceedings of the 24th International Conference on World Wide Web, pp. 111–112. ACM (2015)
37. Bi, F., Wu, D.: A proximal alternating-direction-method-of-multipliers-based nonnegative latent factor model. In: 2021 IEEE International Conference on Big Knowledge (ICBK), pp. 353–360. https://doi.org/10.1109/ICKG52313.2021.00054
38. Wang, Q., Peng, B., Shi, X., Shang, T., Shang, M.: DCCR: deep collaborative conjunctive recommender for rating prediction. IEEE Access. **7**, 60186–60198 (2019)
39. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **1**, 1–30 (2006)
40. Rosner, B., Glynn, R.J., Lee, M.L.T.: The Wilcoxon signed rank test for paired comparisons of clustered data. Biometrics. **62**(1), 185–192 (2006)
41. Wu, D., Luo, X., Wang, G., Shang, M., Yuan, Y., Yan, H.: A highly accurate framework for self-labeled semisupervised classification in industrial applications. IEEE Trans. Ind. Inform. **14**(3), 909–920 (2018)
42. Zhou, Z., Feng, J.: Deep forest: towards an alternative to deep neural networks. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI, pp. 3553–3559 (2017)
43. Ren, X., Song, M., Haihong, E., Song, J.: Context-aware probabilistic matrix factorization modeling for point-of-interest recommendation. Neurocomputing. **241**, 38–55 (2017)
44. Liu, Z., Luo, X., Wang, Z.: Convergence analysis of single latent factor-dependent, nonnegative, and multiplicative update-based nonnegative latent factor models. IEEE Trans. Neural Netw. Learn Syst. **32**(4), 1737–1749 (2020)
45. Wang, C., Liu, Q., Wu, R., Chen, E., Liu, C., Huang, X., Huang, Z.: Confidence-aware matrix factorization for recommender systems. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, pp. 434–442 (2018)
46. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Zhou, M.: A deep latent factor model for high-dimensional and sparse matrices in recommender systems. IEEE Trans. Syst. Man Cybern. Syst. **51**(7), 4285–4296 (2021)
47. Song, Y., Li, M., Luo, X., Yang, G., Wang, C.: Improved symmetric and nonnegative matrix factorization models for undirected, sparse and large-scaled networks: a triple factorization-based approach. IEEE Trans. Ind. Inform. **16**(5), 3006–3017 (2020)
48. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.-S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, pp. 173–182 (2017)
49. Li, P., Wang, Z., Ren, Z., Bing, L., Lam, W.: Neural rating regression with abstractive tips generation for recommendation. In: Proceedings of the 40th ACM International Conference on Research and Development in Information Retrieval, SIGIR, pp. 345–354 (2017)
50. He, X., Chua, T.-S.: Neural factorization machines for sparse predictive analytics. In: Proceedings of the 40th ACM International Conference on Research and Development in Information Retrieval, SIGIR, pp. 355–364 (2017)
51. Wu, D., Luo, X.: Robust latent factor analysis for precise representation of high-dimensional and sparse data. IEEE/CAA J. Autom. Sin. **8**(4), 796–805 (2021)

Chapter 4
Improving Robustness of Latent Feature Learning Using L_1-Norm

4.1 Overview

In this era of information explosion, big data are generated from various industrial applications [1–4]. Realizing intelligent recommender systems (RSs) to filter the required information is a very challenging problem [5, 6]. Up to now, various methods have been proposed to implement an RS, among which collaborative filtering (CF) is very popular [7–13].

Great efforts have been made to implement various CF-based models, among which a latent feature learning (LFL) model [1, 14, 15] is widely used because of its efficient and scalable industrial applications. For example, an LFL model is developed based on the user-item rating matrix [3, 16–18]. Please note that users cannot evaluate all products in a real scenario such as Amazon [19] because the number of products is extremely large. Therefore, the user-item matrix is usually high dimensional and incomplete (HDI).

Given an HDI matrix, an LFL model maps users and items into the same low-dimensional latent feature space to train the low-rank matrices based on observed items only [9]. It is crucial to design an objective function of LFL model that usually has the form of *Loss* + *Penalty* [25–27].

Considering the *Loss* of an LFL model usually depends on the L_1 or L_2 norm defined on the known data of the HDI matrix. Figure 4.1 shows the difference between L_1 norm-oriented and L_2 norm-oriented *Losses*:

(a) The L_1-norm is less sensitive to outliers than L_2-norm, which boosts the robustness of a related model [20–24].
(b) The L_2-norm is smoother than the L_1-norm when the predictions and ground truth data are close, which enhances the stability of a related model, as shown in Fig. 4.1(b) [25].

Since realistically generated HDI matrices usually have extremely high or very low outliers (for example, random scores filled by careless/malicious users) [26, 27], this

D. Wu, *Robust Latent Feature Learning for Incomplete Big Data*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-19-8140-1_4

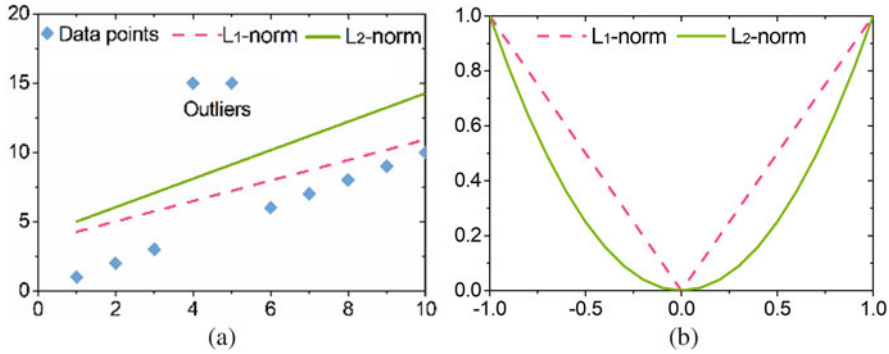


Fig. 4.1 Differences between L_1-norm and L_2-norm: (a) fitness on ten samples that has two outliers, (b) loss functions with L_1-norm and L_2-norm, respectively. Note that the difference between ground truth and predictions is small (e.g., less than 1)

can greatly damage the performance of LFL models that rely only on L_2 norm-oriented *Loss*. On the other hand, the prediction performance of the LFL model, which only depends on the L_1 norm-oriented *Loss*, is unstable, which may greatly damage its overall prediction accuracy for unknown data. From this point of view, the LFL model with loss dependent on L_1-norm or L_2-norm cannot well describe the HDI matrix [24, 28].

According to the existing research, the *Loss* of an existing LFL model usually depends only on the L_1 norm, L_2 norm, or other distance measures, such as Kullback-Leibler Divergence [25, 26]. Related studies [36–40] also suggest combining L_1 and L_2 norms in the *Penalty* rather than *Loss* of an LFL model. In this section, an L_1-and-L_2 norm-oriented latent feature (L^3F) model is introduced. It is very different from the existing LFL model because of its L_1-and-L_2 norms-oriented *Loss* and adopts an adaptive weighting strategy, which well aggregates the influence of L_1-norm and L_2-norm.

4.2 Related Work

An LFL model is widely adopted for implementing RSs [16, 17]. Up to now, various sophisticated LFL models have been proposed, including a fast-converging [30], a bias-based [17], a probabilistic [29], a unconstrained non-negative [33], a non-negativity-constrained [18, 43], a posterior-neighborhood-regularized [41], a data-aware [32], a general deep [31], a randomized [42], and a data characteristic-aware [34] ones. They all use L_2 norm-oriented *Loss*, making they be highly sensitive to outliers [24, 26]. To make the LFL model less sensitive to outliers, Zhu et al., Suggestion adopting the L_1-norm-oriented *Loss* method [23]. However, the LF model with L_1-norm-oriented *Loss* may have multiple solution spaces because L_1 norm is not as smooth as L_2 norm.

On the other hand, matrix completion [35–38] or feature representation [39, 40] models use L_1-norm and L_2-norm to construct their models, to achieve model sparsity [44] or generality [37]. However, different from the existing models, the L^{3}F model takes into account the influence of multi-criteria guidance loss. It realizes the loss of L_1-norm and L_2-norm, and effectively aggregates L_1-norm and L_2-norm through an adaptive weighting strategy, thus realizing the robustness to outliers and the stability of the model simultaneously.

Recently, the deep neural network (DNN) has attracted much attention of researchers [45–47]. There are many sophisticated DNN-based models, including autoencoder-based [45], multitask learning-oriented [47], and hybrid autoencoder-based [46] ones. However, when processing HDI data, the performance of the DNN-based models is obtained with a higher computational burden, while the L^{3}F model is not subject to this restriction.

4.3 An L_1-and-L_2-Norm-Oriented Latent Feature Model

4.3.1 Objective Formulation

As shown above, either L_1-norm or L_2-norm -oriented *Loss* has its advantage in describing HDI data [24, 28]. To aggregate their effects, it can build an L_1-and-L_2-norm-oriented *Loss* as:

$$\begin{aligned}\arg\min_{X,Y} \varepsilon(X, Y) &= \alpha_1 \left\| \Omega \odot \left(\mathrm{H} - \mathrm{W} - XY^{\mathrm{T}} \right) \right\|_{L_1} \\ &+ \alpha_2 \left\| \Omega \odot \left(\mathrm{H} - \mathrm{W} - XY^{\mathrm{T}} \right) \right\|_{L_2}^2 + \lambda \left(\|X\|_{L_2}^2 + \|Y\|_{L_2}^2 \right),\end{aligned} \tag{4.1}$$

where α_1 and α_2 are aggregation weights controlling the effects of L_1-norm and L_2-norm, respectively. Note that $\alpha_1 + \alpha_2 = 1$ and $\alpha_1, \alpha_2 \geq 0$ [16, 18]:

$$\begin{aligned}&\arg\min_{X,Y} \varepsilon(X, Y) = \\ &\underbrace{\underbrace{\alpha_1 \sum_{(m,\, n) \in H_o} \left| h_{m,n} - w_{m,n} - \sum_{k=1}^{d} x_{m,k} y_{n,k} \right|}_{L_1 - norm - oriented\ Loss} + \underbrace{\alpha_2 \sum_{(m,\, n) \in H_o} \left(h_{m,n} - w_{m,n} - \sum_{k=1}^{d} x_{m,k} y_{n,k} \right)^2}_{L_2 - norm - oriented\ Loss}}_{L_1 - and - L_2 - norm - oriented\ Loss} \\ &+ \underbrace{\lambda \sum_{(m,\, n) \in H_o} \left(\sum_{k=1}^{d} (x_{m,k})^2 + \sum_{k=1}^{d} (y_{n,k})^2 \right)}_{L_2 - norm - oriented\ Penalty},\end{aligned} \tag{4.2}$$

where $x\ m,k$ and $y\ n,k$ denote specific entries in $\mathbf{X}$ and $\mathbf{Y}$, $w_{m,n}$ is given by $w_{m,n} = \mu + b_m + b_n$ as μ denotes the global average of H_o, b_m denotes the observed deviations on *m-th* row, and b_n denotes the observed deviations on *n-th column*, respectively.

Note that when $w_{m,n} = 0$, (4.2) does not include linear bias, which can be regarded as a special case of an L^3F model. an L^3F models with and without linear bias are marked as $L^3F_{b^-}$ and L^3F_b, respectively.

4.3.2 Model Optimization

The instant loss on a single rating $h_{m,n}$ is considered in (4.2) as:

$$\begin{aligned}\varepsilon_{u,i} = \alpha_1 |h_{m,n} - w_{m,n} - \sum_{k=1}^{d} x_{m,k} y_{n,k}| + \alpha_2 \left(h_{m,n} - w_{m,n} - \sum_{k=1}^{d} x_{m,k} y_{n,k} \right)^2 \\ + \lambda \left(\sum_{k=1}^{d} (x_{m,k})^2 + \sum_{k=1}^{d} (y_{n,k})^2 \right).\end{aligned} \tag{4.3}$$

Then, in the t-th iteration, the opposite direction of the stochastic gradient of (4.3) is as follows:

$$\forall k \in \{1, 2, \ldots, d\} : \begin{cases} x_{m,k}^t = x_{m,k}^{t-1} - \eta \dfrac{\partial \varepsilon_{m,n}^{t-1}}{\partial x_{m,k}^{t-1}}, \\ y_{n,k}^t = y_{n,k}^{t-1} - \eta \dfrac{\partial \varepsilon_{m,n}^{t-1}}{\partial y_{n,k}^{t-1}}; \end{cases} \tag{4.4}$$

where $\varepsilon_{m,n}^{t-1}$, $x_{m,k}^{t-1}$, and $y_{n,k}^{t-1}$ denote the states of ε, $x\ m,k$, and $y\ n,k$ during the $(t-1)$th iteration, and η denotes the learning rate, respectively. Let $\Delta_{m,n} = h_{m,k} - w_{m,n} - \sum_{k=1}^{d} x_{m,k} y_{n,k}$, then the expression of (4.3) depends on the sign of $\Delta_{m,n}$:

$$\begin{cases} \Delta_{m,n}^{t-1} \geq 0 : \begin{cases} \varepsilon_{m,n}^{t-1} = \alpha_1^{t-1} \Delta_{m,n}^{t-1} + \alpha_2^{t-1} \left(\Delta_{m,n}^{t-1} \right)^2 \\ + \lambda \left(\sum_{k=1}^{d} \left(x_{m,k}^{t-1} \right)^2 + \sum_{k=1}^{d} \left(y_{n,k}^{t-1} \right)^2 \right) \end{cases} \\ \Delta_{m,n}^{t-1} < 0 : \begin{cases} \varepsilon_{m,n}^{t-1} = -\alpha_1^{t-1} \Delta_{m,n}^{t-1} + \alpha_2^{t-1} \left(\Delta_{m,n}^{t-1} \right)^2 \\ + \lambda \left(\sum_{k=1}^{d} \left(x_{m,k}^{t-1} \right)^2 + \sum_{k=1}^{d} \left(y_{n,k}^{t-1} \right)^2 \right) \end{cases} \end{cases}, \tag{4.5}$$

where α_1^{t-1}, α_2^{t-1} and $\Delta_{m,n}^{t-1}$ are the states of α 1, α 2 and $\Delta_{m,n}$ during the $(t-1)$th iteration, respectively.

By combining (4.4) and (4.5) to have the following scheme:

$$\text{On } h_{m,n}, \forall k \in \{1, 2, \ldots, d\}:$$
$$\begin{cases} \Delta_{m,n}^{t-1} > 0: \begin{cases} x_{m,k}^t = x_{m,k}^{t-1} + \alpha_1^{t-1}\eta y_{n,k}^{t-1} + \alpha_2^{t-1}\eta y_{n,k}^{t-1}\Delta_{m,n}^{t-1} - \eta\lambda x_{m,k}^{t-1} \\ y_{n,k}^t = y_{n,k}^{t-1} + \alpha_1^{t-1}\eta x_{m,k}^{t-1} + \alpha_2^{t-1}\eta x_{m,k}^{t-1}\Delta_{m,n}^{t-1} - \eta\lambda y_{n,k}^{t-1} \end{cases} \\ \Delta_{m,n}^{t-1} < 0: \begin{cases} x_{m,k}^t = x_{m,k}^{t-1} - \alpha_1^{t-1}\eta y_{n,k}^{t-1} + \alpha_2^{t-1}\eta y_{n,k}^{t-1}\Delta_{m,n}^{t-1} - \eta\lambda x_{m,k}^{t-1} \\ y_{n,k}^t = y_{n,k}^{t-1} - \alpha_1^{t-1}\eta x_{m,k}^{t-1} + \alpha_2^{t-1}\eta x_{m,k}^{t-1}\Delta_{m,n}^{t-1} - \eta\lambda y_{n,k}^{t-1} \end{cases} \end{cases}. \tag{4.6}$$

4.3.3 Self-Adaptive Aggregation

To finely aggregate the effects of L_1-norm and L_2-norm -oriented *Losses*,α_1 and α_2 are made adaptive according to the training error. Let *Lt* 1 and *Lt* 2 denote the partial loss depending on L_1-norm and L_2-norm in (4.2) at the t-th iteration, respectively. The main idea is to increase α_1 and decrease α_2 if *Lt* 1 $<$ *Lt* 2, and decrease α_1 and increase α_2 otherwise.

Definition 4.1 Let *Lt* 1 and *Lt* 2 be the states of partial *Losses* separately depending on L_1-norm and L_2-norm in (4.2) at the t-th training iteration, then *Lt* 1 and *Lt* 2 are given as follows:

$$\begin{aligned} &\forall m \in \{1, 2, \ldots, |M|\}, \forall n \in \{1, 2, \ldots, |N|\}: \\ &L_1^t = \sum_{(m,\ n)\in H_o} |\Delta_{m,n}^t|, L_2^t = \sum_{(m,\ n)\in H_o} \left(\Delta_{m,n}^t\right)^2. \end{aligned} \tag{4.7}$$

Definition 4.2 Let *Lt* 12 be the state of L^{3}F's *Loss* based on aggregating *Lt* 1 and *Lt* 2 in the t-th training iteration, then *Lt* 12 is formulated as follows:

$$\begin{aligned} &\forall m \in \{1, 2, \ldots, |M|\}, \forall n \in \{1, 2, \ldots, |N|\}: \\ &L_{12}^t = \alpha_1^t \sum_{(m,\ n)\in H_o} |\Delta_{m,n}^t| + \alpha_2^t \sum_{(m,\ n)\in H_o} \left(\Delta_{m,n}^t\right)^2 = \alpha_1^t L_1^t + \alpha_2^t L_2^t. \end{aligned} \tag{4.8}$$

Definition 4.3 Given *Lt* 1, *Lt* 2, and *Lt* 12, let C*t* *L*1, C*t* *L*2, and C*t* *L*12 be the cumulative *Loss* corresponding to *Lt* 1, *Lt* 2 and *Lt* 12 till the t-th iteration, respectively. Then C*t* *L*1, C*t* *L*2, and C*t* *L*12 are given as follows:

$$C_{L_1}^t = \sum_{j=1}^{t} L_1^j, C_{L_2}^t = \sum_{j=1}^{t} L_2^j, C_{L_{12}}^t = \sum_{j=1}^{t} L_{12}^j. \tag{4.9}$$

Based on Definitions 4.1–4.3, Theorem 4.1 is presented as follows.

Theorem 4.1 Considering an L^{3}F model, assuming that its *Lt* 1 and *Lt* 2 lie in the scale of [0, 1]. If αt 1 and αt 2 fulfill the following condition:

$$\alpha_1^t = \frac{e^{-\gamma C_{L_1}^{t-1}}}{e^{-\gamma C_{L_1}^{t-1}} + e^{-\gamma C_{L_2}^{t-1}}}, \alpha_2^t = \frac{e^{-\gamma C_{L_2}^{t-1}}}{e^{-\gamma C_{L_1}^{t-1}} + e^{-\gamma C_{L_2}^{t-1}}}, \tag{4.10}$$

then the following equality holds:

$$C_{L_{12}}^{t_{max}} \leq min\left\{C_{L_1}^{t_{max}}, C_{L_2}^{t_{max}}\right\} + \frac{\ln 2}{\gamma} + \frac{\gamma t_{max}}{8}. \tag{4.11}$$

Note that in Theorem 4.1, αt 1 and αt 2 denote the states of α_1 and α_2 in the t-th training iteration, and γ denotes a hyper-parameter controlling their learning rates. Specifically, the upper bound becomes $min\{Ct_{max}\ L1,\ Ct_{max}\ L2\}$ + $\ln 2\sqrt{\ln t_{max}}$ $+t_{max}/(8\sqrt{\ln t_{max}})$, with $\gamma = \sqrt{1/\ln t_{max}}$. Note that the term $\ln 2\sqrt{\ln t_{max}} + t_{max}/\left(8\sqrt{\ln t_{max}}\right)$ is linearly bounded by the number of iterations. Hence, Ct_{max} $L12$ is comparable to the minimum of Ct_{max} $L1$ and Ct_{max} $L2$ after t_{max} training iterations. Based on Theorem 4.1, it has the following proposition:

Proposition 4.1 Given that $\gamma = \sqrt{1/\ln t_{max}}$, if $Ct_{max}\ L1 > Ct_{max}\ L2$, the following inequality holds:

$$C_{L_{12}}^{t_{max}} \leq C_{L_2}^{t_{max}} + const < C_{L_1}^{t_{max}} + const, \tag{4.12}$$

Otherwise if $Ct_{max}\ L1 < Ct_{max}\ L2$, the following inequality holds:

$$C_{L_{12}}^{t_{max}} \leq C_{L_1}^{t_{max}} + const < C_{L_2}^{t_{max}} + const, \tag{4.13}$$

where $\lim t_{max} \to_{\infty} const = 19.45$.

Remark 4.1 Proposition 4.1 states that Ct_{max} $L12$ is bounded by Ct_{max} $L1$ + *const* and Ct_{max} $L2$ + *const* with the condition of $\gamma = \sqrt{1/\ln t_{max}}$. Hence, an L^{3}F model's cumulative prediction error is always comparable to (or not larger than) that of an LFL model solely built on an L_1 or L_2 norm-oriented *Loss* during the training process.

Note that the time complexity of L^{3}F is $\Theta(t_{max} \times |H_O| \times d)$, where both t_{max} and d are positive constants [48].

4.4 Performance Analysis

4.4.1 General Settings

Datasets Eight benchmark datasets are selected for experiments. Table 4.1 summarizes their properties. In detail, 80%–20% of train-test settings are adopted.

Baseline Table 4.2 gives a brief description of comparison models. To evaluate prediction accuracy, mean absolute error (MAE) and root mean squared error (RMSE) are computed.

Table 4.1 Summary of experimental datasets

No.	Name	|M|	|N|	$\|H_o\|$	Density (%)
D1	Dating	135,359	168,791	17,359,346	0.08
D2	Douban	129,490	58,541	16,830,839	0.22
D3	Eachmovie	72,916	1628	2,811,718	2.37
D4	Epinion	755,760	120,492	13,668,321	0.02
D5	Flixter	147,612	48,794	8,196,077	0.11
D6	Jester	24,983	100	1,186,324	47.49
D7	MovieLens_10M	71,567	65,133	10,000,054	0.21
D8	MovieLens_20M	138,493	26,744	20,000,263	0.54
D9	Yahoo-R2	200,000	136,736	76,344,627	0.28

Table 4.2 Summary of compared models

Model	Description
L_1-LF_{b^-}	L_1-LF is a basic LFL model built on an L_1-norm-oriented loss. L_1-LF_{b^-} stands for an L_1-LF model without linear biases.
L_1-LF_b	An L_1-LF model with linear biases.
L_2-LF_{b^-}	A sohposticated LFL model proposed in 2009 [17]. It is built on an L_2-norm-oriented loss solely. L_2-LF_{b^-} stands for an L_2-LF model without linear biases..
L_2-LF_b	An L_2-LFL model with linear biases.
NLF	A sophisticated LFL model was proposed in 2016 [18]. It improves an L_2-LF model by introducing the non-negative constraints into it for efficiently describing non-negative data.
FNLF	A fast non-negative LFL model based on a generalized momentum method [9]. It is proposed in 2018 [9] and also relies on an L_2-norm-oriented loss solely.
AutoRec	A autoencoder-based model was proposed in 2015 [45]. It is a representative DNN-based recommender model.
NRT	A DNN-based model proposed in 2017 [46]. It adopts a multi-task learning-incorporated framework based on multilayered perceptron and recurrent neural networks.
DCCR	A DNN-based model proposed in 2019 [47]. It improves AutoRec with the consideration of two different networks.
$L^3F_{b^-}$	An L^3F model without linear biases [48].
L^3F_b	An L^3F model with linear biases [48].

4.4.2 L^3F's Aggregation Effects

This section empirically studies how L^3F achieves balanced aggregation effects between L_1 and L_2 norm-oriented *Losses* from two aspects when training.

Monitoring the changes of α_1 and α_2. The results on D1 are presented in Fig. 4.2. From them, it can be found that during the training process of $L^3F_{\bar{b}}$/L^3F_b, both α_1 and α_2 adaptively change first and then converge to the different constants on different datasets. This phenomenon indicates that L^3F adaptively controls the aggregation effects of L_1 and L_2 norm-oriented *Losses*.

Testing L^3F's rating prediction accuracy. L^3F is compared with LFL models built on an L_1 or L_2 norm-oriented *Loss* solely. Figure 4.3 presents the comparison results of models without linear biases on D1.

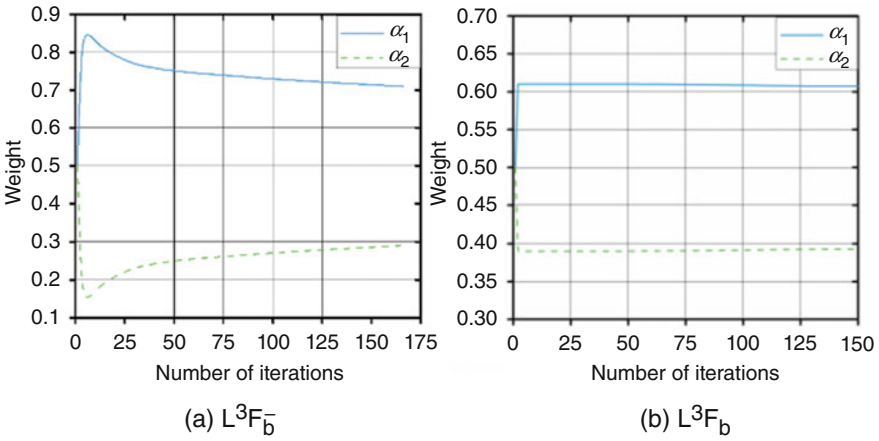


Fig. 4.2 The changes of α_1 and α_2 of L^3F during the training process on D1

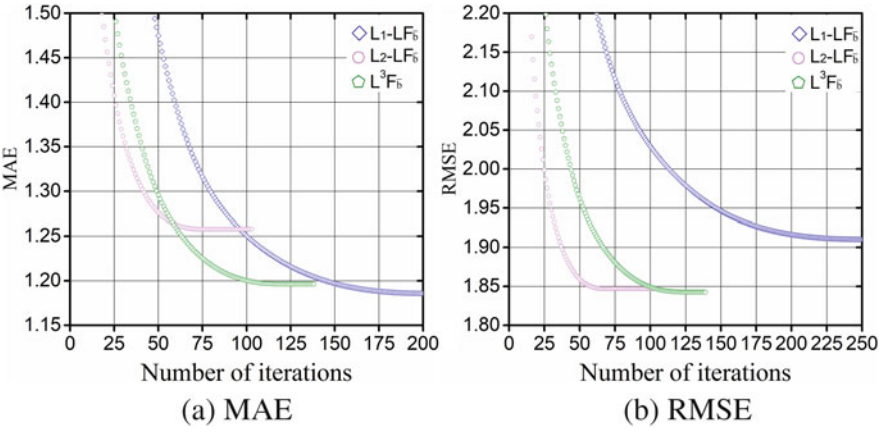


Fig. 4.3 The training process of $L^3F_{\bar{b}}$, L_1-$LF_{\bar{b}}$, and L_2-$LF_{\bar{b}}$ on D1

4.4.3 Comparison Between L^3F and Baselines

To draw fair comparisons, the following settings are used: (a) setting $d = 20$ for all the LF-based models, (b) fivefold cross-validations are adopted and report the average results, (c) the other hyper-parameters are adjusted on onefold of each dataset to achieve the best performance of each model and then adopting the same values on the remaining fourfolds.

4.4.3.1 Comparison of Rating Prediction Accuracy

To check whether $L^3F_ь/L^3F_b$ has a higher rating prediction accuracy than the other models, statistical analysis is also conducted in Table 4.3, i.e., the win/tie/loss counts of $L^3F_ь/L^3F_b$ versus other models one by one and the Friedman test. The comparison results verify that $L^3F_ь/L^3F_b$ achieves a higher rating prediction accuracy than the other models. Further, L^3F_b achieves a smaller F-rank value than $L^3F_{\bar{b}}$, which denotes that L^3F_b outperforms $L^3F_{\bar{b}}$ in general.

4.4.3.2 Comparison of Computational Efficiency

Figure 4.4 presents the CPU running time of all the models when training for rating prediction. The results show that L^3F's computational efficiency is comparable to that of LFL models and is higher than that of DNNs-based models.

4.4.4 L^3F's Robustness to Outlier Data

Artificial Dataset An artificial HDI matrix is generated with the following characteristics to evaluate L^3F's sensitivity to outlier data: (a) it has 5000 users (rows) and 1000 items (columns), (b) its known entries take 2% of the whole entry set only, which are generated at random in the range of [0, 1]. Afterward, outlier users who randomly pick a subset of items to assign them with the same maximum or minimum rating are gradually added to this dataset. The percentage of outlier users increases from 0% to 200% with an interval of 10%. Note that the outlier users are added to the training set. The tests compare $L^3F_ь/L^3F_b$ with L_1-$LF_ь/L_1$-LF_b and L_2-$LF_ь/L_2$-LF_b. Figure 4.5 depicts the comparison results of models without linear biases.

Table 4.3 The comparison results including win/tie/loss counts statistic and Friedman test

Dataset	Metric	L_1-LF_{b^-}	L_1-LF_b	L_2-LF_{b^-}	L_2-LF_b	NLF	FNLF	AutoRec	NRT	DCCR	$L^3F_{b^-}$	L^3F_b
D1	MAE	1.1606	1.1492	1.2392○●	1.2328○●	1.2617○●	1.2588○●	1.2610○●	1.2303○●	1.2574○●	1.1740	1.1781
	RMSE	1.8672○●	1.8025○●	1.8066○●	1.7809●	1.8245○●	1.8215○●	1.8027○●	1.8101○●	1.8013○●	1.7987	1.7704
D2	MAE	0.5394●	0.5245	0.5537○●	0.5516○●	0.5590○●	0.5592○●	0.5606○●	0.5675○●	0.5581○●	0.5412	0.5347
	RMSE	0.7344○●	0.7222○●	0.7139●	0.6993	0.7150○●	0.7139●	0.7080●	0.7106○●	0.7074●	0.7145	0.7000
D3	MAE	0.1713○●	0.1705○●	0.1732○●	0.1730○●	0.1767○●	0.1763○●	0.1784○●	0.1774○●	0.1775○●	0.1700	0.1698
	RMSE	0.2290○●	0.2278○●	0.2251○	0.2262○●	0.2264○●	0.2259○●	0.2305○●	0.2301○●	0.2289○●	0.2249	0.2256
D4	MAE	0.2517○●	0.2223	0.3011○●	0.2938○●	0.3047○●	0.3036○●	0.3014○●	0.3002○●	0.3036○●	0.2512	0.2422
	RMSE	0.6100○●	0.4875●	0.5958●	0.4821●	0.5994○●	0.5977○●	0.5946●	0.5926●	0.5952●	0.5969	0.4775
D5	MAE	0.6319○●	0.6042	0.6447○●	0.6207●	0.6550○●	0.6520○●	0.6295●	0.6337○●	0.6308●	0.6318	0.6050
	RMSE	0.9098○●	0.8450●	0.8961○●	0.8383●	0.9056○●	0.9038○●	0.8682●	0.8677●	0.8792●	0.8960	0.8326
D6	MAE	0.7580○●	0.7600○●	0.7664○●	0.7676○●	0.7769○●	0.7778○●	0.7905○●	0.7878○●	0.7883○●	0.7562	0.7578
	RMSE	1.0206○●	1.0177○●	0.9957●	0.9982○●	1.0049○●	1.0003○●	1.0078○●	1.0056○●	1.0042○●	0.9978	0.9935
D7	MAE	0.5986○●	0.6038○●	0.5999○●	0.6067○●	0.6080○●	0.6068○●	0.6048○●	0.6035○●	0.6002○●	0.5956	0.5981
	RMSE	0.7953○●	0.8048○●	0.7819	0.7936○●	0.7893○	0.7881○	0.7865○	0.7834○	0.7847○	0.7821	0.7899
D8	MAE	0.5877○●	0.5927○●	0.5886○●	0.5955○●	0.5977○●	0.5961○●	0.5947○●	0.6018○●	0.5902○●	0.5848	0.5863
	RMSE	0.7873○●	0.7960○●	0.7737	0.7848○●	0.7819○●	0.7798○	0.7802○	0.7798○	0.7789○	0.7743	0.7806
D9	MAE	0.7661	0.7916○●	0.8037○●	0.8475○●	0.8342○●	0.8228○●	0.8029○●	0.8221○●	0.8058○●	0.7742	0.7840
	RMSE	1.1129○●	1.1460○●	1.0596	1.0968○●	1.0746○	1.0704○	1.0629○	1.1150○●	1.0931○●	1.0601	1.0776
Statistical analysis	○win/ tie/loss	15/0/3	12/0/6	12/0/6	13/0/5	18/0/0	17/0/1	14/0/4	16/0/2	14/0/4	–	–
	●win/ tie/loss	16/0/2	14/0/4	14/0/4	17/0/1	16/0/2	15/0/3	15/0/3	16/0/2	16/0/2	–	–
	F-rank[a]	6.83	5.50	4.97	5.56	8.83	7.64	7.11	7.19	6.31	3.28	2.78

a The smaller F-rank value denotes a higher rating prediction accuracy

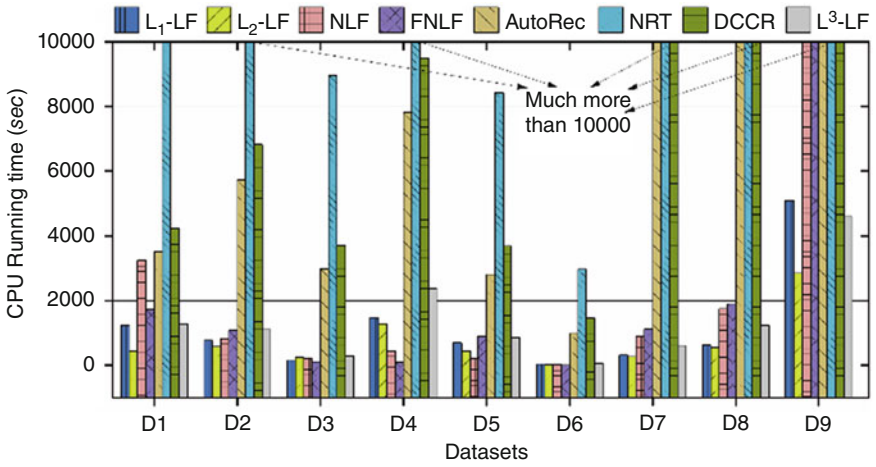


Fig. 4.4 The comparison CPU running time of involved models on D1–D9

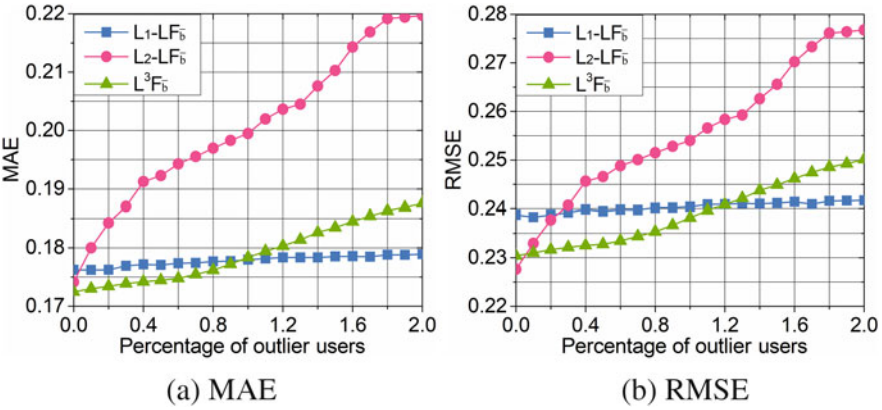


Fig. 4.5 The outlier data sensitivity tests of $L^3F_{\bar{b}}$, L_1-$LF_{\bar{b}}$, and L_2-$LF_{\bar{b}}$ on the artificial dataset

4.5 Summary

An L_1-and-L_2 norm-oriented latent feature (L^3F) model is introduced in this chapter. It is very different from the existing LFL model because of its L_1-and-L_2 norms-oriented *Loss* and adopts an adaptive weighting strategy. Empirical studies demonstrate that an L^3F model can well aggregate the merits of L_1-norm and L_2-norm in handling an HDI matrix with outliers. However, an L^3F model's performance is sensitive to its regularization coefficient λ, which is data-dependent and should be tuned carefully. Therefore, its self-adaptation is desired to enhance L^3F's practicability.

References

1. Luo, X., Zhou, M., Li, S., Hu, L., Shang, M.: Non-negativity constrained missing data estimation for high-dimensional and sparse matrices from industrial applications. IEEE Trans. Cybern. **50**(5), 1844–1855 (2020)
2. Luo, X., Zhou, M., Li, S., Shang, M.: An inherently nonnegative latent factor model for high-dimensional and sparse matrices from industrial applications. IEEE Trans. Ind. Inform. **14**(5), 2011–2022 (2018)
3. Wu, D., Zhang, P., He, Y., Luo, X.: A double-space and double-norm ensembled latent factor model for highly accurate web service QoS prediction. In: EEE Transactions on Services Computing, p. 1 (2022)
4. Luo, X., Chen, M., Wu, H., Liu, Z., Yuan, H., Zhou, M.: Adjusting learning depth in nonnegative latent factorization of tensors for accurately modeling temporal patterns in dynamic QoS data. IEEE Trans. Autom. Sci. Eng. **18**(4), 2142–2155 (2021)
5. Luo, X., et al.: Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data. IEEE Trans. Cybern. **48**(4), 1216–1228 (2017)
6. Luo, X., Zhou, M., Wang, Z., Xia, Y., Zhu, Q.: An effective scheme for QoS estimation via alternating direction method-based matrix factorization. IEEE Trans. Serv. Comput. **12**(4), 503–518 (2016)
7. Luo, X., Zhou, M., Xia, Y., Zhu, Q.: An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. IEEE Trans. Ind. Inform. **10**(2), 1273–1284 (2014)
8. Luo, X., et al.: An incremental-and-static-combined scheme for matrix-factorization-based collaborative filtering. IEEE Trans. Autom. Sci. Eng. **13**(1), 333–343 (2016)
9. Luo, X., Liu, Z., Li, S., Shang, M., Wang, Z.: A fast non-negative latent factor model based on generalized momentum method. IEEE Trans. Syst. Man Cybern. Syst. **51**(1), 610–620 (2018)
10. Wu, L., Sun, P., Hong, R., Ge, Y., Wang, M.: Collaborative neural social recommendation. IEEE Trans. Syst. Man Cybern. Syst. **51**, 464–476 (2018)
11. Yu, Z., Wu, D., He, Y.: A robust latent factor analysis model for incomplete data recovery in wireless sensor networks. In: 2022 IEEE International Conference on Edge Computing and Communications (EDGE), pp. 178–183 (2022)
12. Wu, D., Luo, X., He, Y., Zhou, M.: A prediction-sampling-based multilayer-structured latent factor model for accurate representation to high-dimensional and sparse data. IEEE Trans. Neural Netw. Learn. Syst., 1–14 (2022)
13. Wu, D., Jin, L., Luo, X.: Pmlf: Prediction-sampling-based multilayer-structured latent factor analysis. In: 2020 IEEE International Conference on Data Mining (ICDM), pp. 671–680 (2020)
14. Luo, X., Ouyang, Y., Xiong, Z.: Improving neighborhood based collaborative filtering via integrated folksonomy information. Pattern Recogn. Lett. **33**(3), 263–270 (2012)
15. Luo, X., Wu, H., Yuan, H., Zhou, M.: Temporal pattern-aware QoS prediction via biased non-negative latent factorization of tensors. IEEE Trans. Cybern. **50**(5), 1798–1809 (2019)
16. Luo, X., Wang, Z., Shang, M.: An instance-frequency-weighted regularization scheme for non-negative latent factor analysis on high-dimensional and sparse data. IEEE Trans. Syst. Man Cybern. Syst. **51**(6), 1–11 (2019)
17. Luo, X., Sun, J., Wang, Z., Li, S., Shang, M.: Symmetric and nonnegative latent factor models for undirected, high-dimensional, and sparse networks in industrial applications. IEEE Trans. Ind. Inform. **13**(6), 3098–3107 (2017)
18. Luo, X., Zhou, M., Li, S., You, Z., Xia, Y., Zhu, Q.: A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method. IEEE Trans. Neural Netw. Learn. Syst. **27**(3), 579–592 (2016)
19. Smith, B., Linden, G.: Two decades of recommender systems at Amazon.com. IEEE Internet Comput. **21**(3), 12–18 (2017)
20. Yuan, Y., Luo, X., Shang, M.-S.: Effects of preprocessing and training biases in latent factor models for recommender systems. Neurocomputing. **275**, 2019–2030 (2018)

21. Ma, W., Li, N., Li, Y., Duan, J., Chen, B.: Sparse normalized least mean absolute deviation algorithm based on unbiasedness criterion for system identification with noisy input. IEEE Access. **6**, 14379–14388 (2018)
22. Ke, Q., Kanade, T.: Robust Subspace Computation Using L1 Norm: School of Computer Science. Carnegie Mellon University, Pittsburgh, PA (2003)
23. Zhu, X., Jing, X.-Y., Wu, D., He, Z., Cao, J., Yue, D., Wang, L.: Similarity-maintaining privacy preservation and location-aware low-rank matrix factorization for QoS prediction based web service recommendation. IEEE Trans. Serv. Comput. **14**(3), 889–902 (2018)
24. Wang, L., Gordon, M.D., Zhu, J.: Regularized least absolute deviations regression and an efficient algorithm for parameter tuning. In: Proceedings of the 6th IEEE International Conference on Data Mining (ICDM), pp. 690–700 (2006)
25. Koenker, R., Hallock, K.F.: Quantile regression. J. Econ. Perspect. **15**(4), 143–156 (2001)
26. Lakshminarayanan, B., Bouchard, G., Archambeau, C.: Robust Bayesian matrix factorisation. In: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, pp. 425–433 (2011)
27. Luo, X., Zhou, Y., Liu, Z., Zhou, M.: Fast and accurate non-negative latent factor analysis on high-dimensional and sparse matrices in recommender systems. In: IEEE Transactions on Knowledge and Data Engineering, p. 1 (2021)
28. Goldstein, T., Osher, S.: The split Bregman method for L1-regularized problems. SIAM J. Imaging Sci. **2**(2), 323–343 (2009)
29. Ren, X., Song, M., Haihong, E., Song, J.: Context-aware probabilistic matrix factorization modeling for point-of-interest recommendation. Neurocomputing. **241**, 38–55 (2017)
30. Yuan, Y., Luo, X., Shang, M., Wu, D.: A generalized and fast-converging non-negative latent factor model for predicting user preferences in recommender systems. In: Proceedings of The Web Conference 2020. Association for Computing Machinery, pp. 498–507 (2020)
31. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Zhou, M.: A deep latent factor model for high-dimensional and sparse matrices in recommender systems. IEEE Trans. Syst. Man Cybern. Syst. **51**(7), 4285–4296 (2021)
32. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Wu, X.: A data-aware latent factor model for web service QoS prediction. In: Advances in Knowledge Discovery and Data Mining, pp. 384–399. Springer, Cham (2019)
33. Luo, X., Zhou, M., Li, S., Wu, D., Liu, Z., Shang, M.: Algorithms of unconstrained non-negative latent factor analysis for recommender systems. IEEE Trans. Big Data. **7**(1), 227–240 (2021)
34. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Wu, X.: A data-characteristic-aware latent factor model for web service QoS prediction. IEEE Trans. Knowl. Data Eng. **34**(6), 2525–2538 (2022)
35. Luo, X., Zhou, Y., Liu, Z., Zhou, M.: Fast and accurate non-negative latent factor analysis on high-dimensional and sparse matrices in recommender systems. IEEE Trans. Knowl. Data Eng., 1–1 (2021)
36. Cherapanamjeri, Y., Gupta, K., Jain, P.: Nearly optimal robust matrix completion. In: Proceedings of the 34th International Conference on Machine Learning, JMLR, pp. 797–805 (2017)
37. Klopp, K.L., Tsybakov, A.B.: Robust matrix completion. Probab. Theory Relat. Fields. **169**(1–2), 523–564 (2017)
38. Mansour, H., Tian, D., Vetro, A.: Factorized robust matrix completion. In: Handbook of Robust Low-Rank and Sparse Matrix Decomposition: Applications in Image and Video Processing, vol. 5. CRC Press, Boca Raton, FL (2016)
39. He, Y., Wu, B., Wu, D., Wu, X.: On partial multi-task learning. In: ECAI 2020, pp. 1174–1181 (2020)
40. Han, Y., Yang, Y., Zhou, X.: Co-regularized ensemble for feature selection. In: Proceedings of 23rd International Joint Conference on Artificial Intelligence (IJCAI), pp. 1380–1386 (2013)

41. Wu, D., He, Q., Luo, X., Shang, M., He, Y., Wang, G.: A posterior-neighborhood-regularized latent factor model for highly accurate web service qos prediction. IEEE Trans. Serv. Comput. **15**(2), 793–805 (2022)
42. Shang, M., Luo, X., Liu, Z., Chen, J., Yuan, Y., Zhou, M.: Randomized latent factor model for high-dimensional and sparse matrices from industrial applications. IEEE/CAA J. Autom. Sin. **6**(1), 131–141 (2019)
43. Xin, L., Yuan, Y., Zhou, M., Liu, Z., Shang, M.: Non-negative latent factor model based on β-divergence for recommender systems. IEEE Trans. Syst. Man Cybern. Syst. **51**(8), 4612–4623 (2019)
44. Shi, Q., Lu, H., Cheung, Y.-M.: Rank-one matrix completion with automatic rank estimation via L1-norm regularization. IEEE Trans. Neural Netw. Learn. Syst. **29**(10), 4744–4757 (2017)
45. Sedhain, S., Menon, A.K., Sanner, S., Xie, L.: AutoRec: Autoencoders meet collaborative filtering. In: Proceedings of the 24th International Conference on World Wide Web, pp. 111–112. ACM (2015)
46. Li, P., Wang, Z., Ren, Z., Bing, L., Lam, W.: Neural rating regression with abstractive tips generation for recommendation. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 345–354 (2017)
47. Wang, Q., Peng, B., Shi, X., Shang, T., Shang, M.: DCCR: deep collaborative conjunctive recommender for rating prediction. IEEE Access. **7**, 60186–60198 (2019)
48. Wu, D., Shang, M., Luo, X., Wang, Z.: An L_1-and-L_2-norm-oriented latent factor model for recommender systems. IEEE Trans. Neural Netw. Learn. Syst., 1–14 (2021)

Chapter 5
Improve Robustness of Latent Feature Learning Using Double-Space

5.1 Overview

In a high dimensional and incomplete (HDI) matrix, the original data is sparse. Among numerous missing data estimation approaches [1–12], latent feature learning (LFL) is widely studied and adopted because of its high efficiency and scalability.

Given an HDI matrix, the predictor-based LFL aims to minimize the differences between the observed data and predictions by training two latent feature matrices, then achieving the low-rank approximation. Although there are many proposed predictors based on LFL [13–27], they have a common characteristic, i.e., the above differences are minimized on the inner product space with L_2-*norm*-oriented *Loss*. While due to the limitations of the inner product space and L_2-norm [15, 28, 29], the predictor based on LFL cannot represent the target HDI matrix characteristics and make accurate predictions. The limitations will be introduced as follows.

In inner product space, although LFL can discover the global similarity among instances and items, it often ignores the fine-grained similarity [28, 29]. Comparatively, distance space can discover the local similarity easily. The difference between inner product space and distance space is shown in Fig. 5.1.

Example 5.1 Given an HDI matrix $\mathbf{H}$ with three instances m_1, m_2, m_3, and two items n_1, n_2, the invocation between instances m and items n is represented as $h_{\mathrm{m,n}}$. e.g., $h_{1,1} = 2$. From $\mathbf{H}$, as $h_{1,1} = h_{3,1} = 2$ and $h_{2,2} = h_{3,2} = 2$, it can find that h_1 and h_2 are similar to h_3, that shows that h_1, h_2 and h_3 are all similar. The similarity is found from all known data of $\mathbf{H}$, so it can be called global similarity. Then it is easy to deduce then n_1, n_2 are similar as $h_{3,1} = h_{3,2} = 2$. The similarity between n_1 and n_2 is called as local similarity because it is discovered from m_3's invocations only. Inner product space use the inner product of two latent feature vectors $h_{m,n} = x_m y_n$ to represent $\mathbf{H}$'s invocations, while distance space use Euclidean distance $h_{m,n} = \|x_m - y_n\|_2$ to represent the invocations. Note, x_m is a row vector of $\mathbf{X}$ corresponding to a instance m, y_n is a column vector of $\mathbf{Y}$ corresponding to an item n, and $\mathbf{X}$, $\mathbf{Y}$ are instance and item latent feature matrix. The representation results are visualized on

D. Wu, *Robust Latent Feature Learning for Incomplete Big Data*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-19-8140-1_5

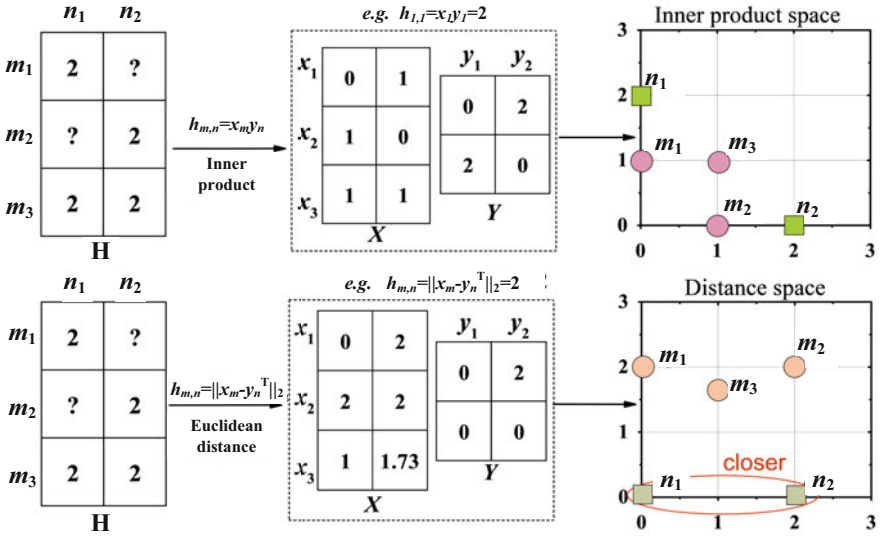


Fig. 5.1 An example that shows the distribution differences of latent features between inner product space and Euclidean distance space in representing an HDI matrix

two-dimensional distribution maps. In inner product space, as m_1, m_2 and m_3 are close to each other, the global similarity of them can be represented well, while the n_1, n_2 are far away from each other, the local similarity between them cannot be represented well. While in distance space, n_1, n_2 become closer so that the local similarity can be better represented.

In L_2-norm, when the ground truths and predicted ones are close to each other, *L_2-norm* is smooth [15]. Hence, the predictor based on *L_2-norm*-oriented *Loss* function can usually obtain stable predictions. However, when HDI matrix is mixed with outlier data, the robustness of the predictor cannot be guaranteed as L_2-norm is sensitive to outliers. Comparatively, *L_1-norm* is less sensitive to outliers than *L_2-norm* [15, 30]. so incorporating *L_1-norm* into the predictor's *Loss* function may improve the robustness greatly.

Example 5.2 In an HDI matrix, the differences between ground truths and predicted ones $\Delta_{m,n}$ can be measured by Loss function. The *L_1-norm*-oriented Loss is $Loss = |\Delta_{m,n}|$,while *L_2-norm*-oriented is $(\Delta_{m,n})^2$. Their functional relationships are shown in Fig.5.2(a), which shows that *L_1-norm* is less sensitive to $\Delta_{m,n}$ than L_2-norm, so *L_1-norm*-oriented *Loss* is more robust to outliers in regressing the ground truth HDI data and depicted in Fig.5.2(b).

The above analyses show that only adopting inner product space and *L_2-norm*-oriented *Loss* function is not the best choice. To multiple advantages of inner product space, distance space, *L_1-norm*, and *L_2-norm*, a new method Double-space and Double-norm Ensembled Latent Feature (D^2E-LF) model was proposed in [22]. Its main ideas are as follows: (1)Double-space. Employing inner product space and

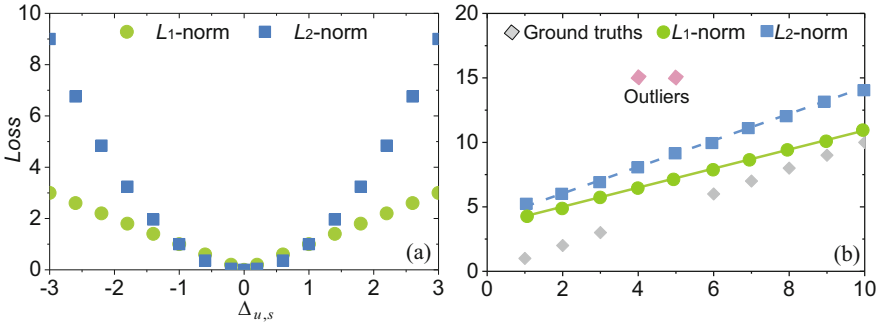


Fig. 5.2 An example: the difference between L_1-norm and L_2-norm in formulating the Loss function of an LFL-based QoS Predictor

distance space to model two kinds of HDI data predictors. (2)Double-norm. Both of these two predictors adopt an *L_1-and-L_2-norm*-oriented Loss function. (3) Ensembled. Reconstructing these two predictor by a weighting strategy.

5.2 Related Work

The most common approach to HDI data prediction is collaborative filtering (CF) [31–36]. Among many model-based CF methods, latent feature learning (LFL)-based is widely popular and investigated for its high efficiency and scalability [37–53]. So far, various sophisticated LFL-based models have been proposed, including data-characteristic-aware [13], non-negative constraint [44], neighborhood integrated [54], data transformation and adaptive weights based [55], posterior-neighborhood-regularized [14], covering-based and neighborhood-aware [57], and generative probabilistic learning framework-based [56] ones. Besides, some LFL-based approaches improve their accuracy by considering the additional geographical information [1] because users in the same area tend to possess similar invoking environments [54].

Compared with the above-mentioned models, the D^{2}E-LF model owns its significance: the current LFL-based methods are developed on the inner product space, which only has the *L_2-norm*-oriented *Loss* oriented, cannot fully represent the HDI data as the inner product space and the *L_2-norm* have their own limitations.

5.3 A Double-Space and Double-Norm Ensembled Latent Feature Model

Motivated by the above two examples depicted in Figs. 5.1 and 5.2, the multi-merits originating from inner product space, distance space, *L_1-norm*, and *L_2-norm* into D^{2}E-LF are integrated by the following three steps:

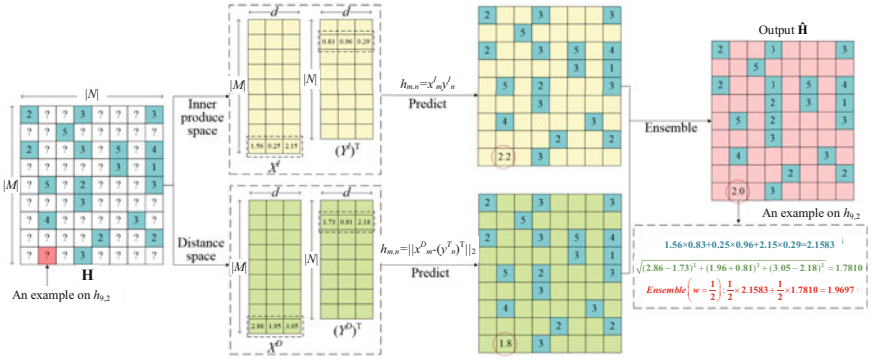


Fig. 5.3 The architecture of the proposed D2E-LF model

Employing HDI matrix **H** to train two kinds of latent feature predictors on inner product space (yellow color) and distance space (green color) based on X_d of **X**.

According to these two kinds of predictors, the missing data (marked '?') will be predicted, respectively.

Ensembling the result of two predictors by weighting strategy to get the final output $\hat{\mathbf{H}}$.

Given an example to illustrate the main idea of D^{2}E-LF, i.e., predicting the missing data $h_{9,2}$, the predicted result of inner produce space and distance space are 2.1583 and 1.7810, so the final result of $h_{9,2}$ is 1.9697 by weighted the above different results. The detail of D^{2}E-LF is described as follows (Fig. 5.3).

5.3.1 *Predictor Based on Inner Product Space (D^{2}E-LF-1)*

First, the common Loss function [32, 50] incorporated Tikhonov regularization adopts the inner product space with an L_2-*norm* as follows.

$$\arg\min \varepsilon(\mathrm{X},\mathrm{Y}) = \frac{1}{2}\left\|\Omega \odot (\mathrm{H}-\mathrm{XY})\right\|_{L_2}^2 + \frac{\lambda}{2}\left(\|\mathrm{X}\|_{L_2}^2 + \|\mathrm{Y}\|_{L_2}^2\right), \tag{5.1}$$

where $\odot$ represent the Hadamard product and Ω is binary index matrix when $h_{m,n}$ is observed $\Omega_{m,n} = 1$ else $\Omega_{m,n} = 0$. And λ is a hyperparameter of the regularization coefficient. As **H** is an HDI matrix, it could expand (5.1) into a density-oriented form to improve efficiency as follows:

$$\begin{aligned}\arg\min \varepsilon(\mathrm{X},\mathrm{Y}) = &\frac{1}{2}\sum_{h_{u,s}\in H_O}\left(h_{m,n}-\sum_{k=1}^{d}x_{m,k}y_{k,n}\right)^2 \\ &+\frac{\lambda}{2}\sum_{h_{m,n}\in H_K}\left(\sum_{k=1}^{d}x_{m,k}^2+\sum_{k=1}^{d}y_{k,n}^2\right),\end{aligned} \tag{5.2}$$

where $x_{m,k}$ denotes the m-th row and k-th column entry of $\mathbf{X}$, $y_{k,n}$ denotes the k-th row and n-th coloum entry of $\mathbf{Y}$, and Ho denotes the known entry set of $\mathbf{H}$. (5.2) can update $x_{m,k}$ and $y_{k,n}$ by stochastic gradient descent (SGD) [43]. For each entry $h_{m,n}$:

$$for\ h_{m,n} \in H_O, \forall k \in \{1, 2, \ldots, d\} : \begin{cases} x_{m,k} \leftarrow x_{m,k} - \eta \dfrac{\partial \varepsilon_{m,n}(\mathrm{X},\ \mathrm{Y})}{\partial x_{m,k}} \\ y_{k,n} \leftarrow y_{k,n} - \eta \dfrac{\partial \varepsilon_{m,n}(\mathrm{X},\ \mathrm{Y})}{\partial y_{k,u}} \end{cases}, \tag{5.3}$$

where η is the learning rate and $\varepsilon_{m,n}(\mathbf{X},\mathbf{Y})$ is the instant state of (5.2) on a single entry $h_{m,n}$.

According to above analyses, the predictor robustness can be improved by incorporating L_1-*norm* into (5.2) to model a predictor with an L_1-*and*-L_2-*norm*-oriented *Loss* on inner product space as (5.4).

$$\begin{aligned} \arg\min \varepsilon\left(X^I, Y^I\right) = & \underbrace{\frac{1}{2}\alpha_1^I \sum_{h_{m,n} \in H_O} |h_{m,n} - \sum_{k=1}^{d} x_{m,k}^I y_{k,n}^I|}_{L_1\text{-norm-oriented } Loss} + \underbrace{\frac{1}{2}\alpha_2^I \sum_{h_{m,n} \in H_O} \left(h_{m,n} - \sum_{k=1}^{d} x_{m,k}^I y_{k,n}^I\right)^2}_{L_2\text{-norm-oriented } Loss} \\ & + \underbrace{\frac{1}{2}\lambda \sum_{h_{m,n} \in H_O} \left(\sum_{k=1}^{d} \left(x_{m,k}^I\right)^2 + \sum_{k=1}^{d} \left(y_{k,n}^I\right)^2\right)}_{\text{Regularization}} \end{aligned} \tag{5.4}$$

where $\alpha I\ 1$ and $\alpha I\ 2$ denote the coefficients of L_1-*norm*- and L_2-*norm*-oriented *Losses* on inner product space, $xI\ m,k$ denotes the m-th row and k-th column entry of X^I, while $yI\ k,n$ denotes the k-row and n-th column of Y^I. In (5.4), it needs to ensure that $\alpha I\ 1 + \alpha I\ 2 = 1$ and $\alpha I\ 1, \alpha I\ 2 \geq 0$. The bias is important to improve the model representation learning ability of LFL-based predictors. So the bias incorporated into (5.4) is as follows:

$$\begin{aligned} \arg\min \varepsilon\left(X^I, Y^I\right) = & \underbrace{\frac{1}{2}\alpha_1^I \sum_{h_{m,n} \in H_O} |h_{m,n} - \sum_{k=1}^{d} x_{m,k}^I y_{k,n}^I - b_m^I - b_n^I - u|}_{L_1\text{-norm-oriented } Loss} \\ & + \underbrace{\frac{1}{2}\alpha_2^I \sum_{h_{m,n} \in H_O} \left(h_{m,n} - \sum_{k=1}^{d} x_{m,k}^I y_{k,n}^I - b_m^I - b_n^I - u\right)^2}_{L_2\text{-norm-oriented } Loss} \\ & + \underbrace{\frac{1}{2}\lambda \sum_{h_{m,n} \in H_O} \left(\sum_{k=1}^{d} \left(x_{m,k}^I\right)^2 + \sum_{k=1}^{d} \left(y_{k,n}^I\right)^2 + \left(b_m^I\right)^2 + \left(b_n^I\right)^2\right)}_{\text{Regularization}}, \end{aligned} \tag{5.5}$$

where u denotes the global average of H_O, $bI\ m$ and $bI\ n$ denotes the bias of m-th instance and n-th items. As the L_1-*norm*-oriented *Loss* term is not differentiable, it needs to be extended to the form on all single entry $h_{m,n}$:

$$\text{On } h_{m,n}:\ \begin{cases} \Delta^I_{m,n} \geq 0 : \varepsilon_{m,n}(X^I,\ Y^I) = \\ \dfrac{1}{2}\left[\alpha^I_1 \Delta^I_{m,n} + \alpha_2 \left(\Delta^I_{m,n}\right)^2 + \lambda\left(\sum\limits_{k=1}^{d}\left(\left(x^I_{m,k}\right)^2 + \left(y^I_{k,n}\right)^2\right) + \left(b^I_m\right)^2 + \left(b^I_n\right)^2\right)\right] \\ \Delta^I_{m,n} < 0 : \varepsilon_{m,n}(X^I,\ Y^I) = \\ \dfrac{1}{2}\left[-\alpha^I_2 \Delta^I_{m,n} + \alpha_2 \left(\Delta^I_{m,n}\right)^2 + \lambda\left(\sum\limits_{k=1}^{d}\left(\left(x^I_{m,k}\right)^2 + \left(y^I_{k,n}\right)^2\right) + \left(b^I_m\right)^2 + \left(b^I_n\right)^2\right)\right] \end{cases}, \tag{5.6}$$

where the error $\Delta I\ m,n = m_{m,n}\text{-}\sum d\ k = 1(xI\ m,kyI\ k,n)\text{-}bI\ m\text{-}bI\ n\text{-}u$. Then, minimizing (5.6) by SGD:

$$\text{On } h_{m,n}, \forall k \in \{1,\ 2,\ \ldots,\ d\}:\ \begin{cases} \Delta^I_{m,n} \geq 0 : \begin{cases} x^I_{m,k} \leftarrow x^I_{m,k} + \dfrac{1}{2}\alpha^I_1 \eta y^I_{k,n} + \alpha^I_2 \eta y^I_{k,n} \Delta^I_{m,n} - \eta\lambda x^I_{m,k} \\ y^I_{k,n} \leftarrow y^I_{k,n} + \dfrac{1}{2}\alpha^I_1 \eta x^I_{m,k} + \alpha^I_2 \eta x^I_{m,k} \Delta^I_{m.n} - \eta\lambda y^I_{k,n} \\ b^I_m \leftarrow b^I_m + \dfrac{1}{2}\alpha^I_1 \eta + \alpha^I_2 \eta \Delta^I_{m,n} - \eta\lambda b^I_m \\ b^I_n \leftarrow b^I_n + \dfrac{1}{2}\alpha^I_1 \eta + \alpha^I_2 \eta \Delta^I_{m,n} - \eta\lambda b^I_n \end{cases} \\ \Delta^I_{m,n} < 0 : \begin{cases} x^I_{m,k} \leftarrow x^I_{m,k} - \dfrac{1}{2}\alpha^I_1 \eta y^I_{k,n} + \alpha^I_2 \eta y^I_{k,n} \Delta^I_{m,n} - \eta\lambda x^I_{m,k} \\ y^I_{k,n} \leftarrow y^I_{k,n} - \dfrac{1}{2}\alpha^I_1 \eta x^I_{m,k} + \alpha^I_2 \eta x^I_{m,k} \Delta^I_{m,n} - \eta\lambda y^I_{k,n} \\ b^I_m \leftarrow b^I_m - \dfrac{1}{2}\alpha^I_1 \eta + \alpha^I_2 \eta \Delta^I_{m,n} - \eta\lambda b^I_m \\ b^I_n \leftarrow b^I_n - \dfrac{1}{2}\alpha^I_1 \eta + \alpha^I_2 \eta \Delta^I_{m,n} - \eta\lambda b^I_n \end{cases} \end{cases}. \tag{5.7}$$

In (5.7), the $\alpha I\ 1$ and $\alpha I\ 2$ can be self-adaptive as the change of training errors of L_1-*norm*-and-L_2-*norm*-oriented *Losses* [15]. And some related definitions are given as follows.

Definition 5.1 Named the partial Loss of L_1-*norm*- and L_2-*norm*-oriented *Losses* at the i-th training iteration on inner produces space as $LI\ 1(i)$ and $LI\ 2(i)$. Then $LI\ 1(i)$ and $LI\ 2(i)$ can be depicted by formulas:

$$L_1^I(i) = \sum_{h_{m,n} \in H_O} |\Delta_{m,n}^I(i)|, L_2^I(i) = \sum_{h_{m,n} \in H_O} \left(\Delta_{m,n}^I(i)\right)^2 \tag{5.8}$$

where $\Delta I\ m,n(i)$ denotes the state of $\Delta I\ m,n$ at i-th training iteration.

Definition 5.2 At the i-th training iteration, the cumulative *Loss* of $LI\ 1(i)$ and $LI\ 2(i)$ on inner produce space can be represented by $CI\ L1(n)$ and $CI\ L2(n)$, and them can be computed by:

$$C_{L_1}^I(i) = \sum_{j=1}^{n} L_1^I(j), C_{L_2}^I(i) = \sum_{j=1}^{n} L_2^I(j). \tag{5.9}$$

Then α_1 and α_2 can be adjusted adaptively according to the Definitions 5.1 and 5.2. so their i-th training iteration can be represented by follows:

$$\begin{aligned} \alpha_1^I(i) &= \frac{e^{-\gamma^I(i) C_{L_1}^I(i-1)}}{e^{-\gamma^I(i) C_{L_1}^I(i-1)} + e^{-\gamma^I(i) C_{L_2}^I(i-1)}}, \\ \alpha_2^I(i) &= \frac{e^{-\gamma^I(i) C_{L_2}^I(i-1)}}{e^{-\gamma^I(i) C_{L_1}^I(i-1)} + e^{-\gamma^I(i) C_{L_2}^I(i-1)}}, \end{aligned} \tag{5.10}$$

Let $\gamma^I(i)$ denote a balance coefficient, which can be set adaptively at each training iteration.

$$\gamma^I(i) = \frac{1}{C_{L_1}^I(i-1) + C_{L_2}^I(i-1)} \tag{5.11}$$

The above process, which adopts L_1-and-L_2-*norm*-oriented *Loss* on inner space to train an HDI data predictor, can be named D^{2}E-LF-1.

5.3.2 *Predictor on Euclidean Distance Space (D^{2}E-LF-2)*

Different from the inner produce space, D^{2}E-LF-2 can be given a result by computing the Euclidean distance of two latent feature matrices. Similar to (5.4), considering the bias, the Euclidean distance model can be formulated as follows:

$$\arg\min \varepsilon\left(X^D, Y^D\right) = \underbrace{\frac{1}{2}\alpha_1^D \sum_{h_{m,n}\in H_O} \left|h_{m,n} - \sqrt{\sum_{k=1}^{d}\left(x_{m,k}^D - y_{k,n}^D\right)^2} - b_m^D - b_n^D - u\right|}_{L_1\text{-norm-oriented } Loss}$$
$$\underbrace{+\frac{1}{2}\alpha_2^D \sum_{h_{m,n}\in H_O}\left(h_{m,n} - \sqrt{\sum_{k=1}^{d}\left(x_{m,k}^D - y_{k,n}^D\right)^2} - b_m^D - b_n^D - u\right)^2}_{L_2\text{-norm-oriented } Loss}$$
$$\underbrace{+\frac{1}{2}\lambda \sum_{h_{m,n}\in H_O}\left(\sum_{k=1}^{d}\left(x_{m,k}^D\right)^2 + \sum_{k=1}^{d}\left(y_{k,n}^D\right)^2 + \left(b_m^D\right)^2 + \left(b_n^D\right)^2\right)}_{\text{Regularization}} \tag{5.12}$$

where αD 1 and αD 2 is the coefficients of L_1-*norm*- and L_2-*norm*-oriented *Losses* on distance space. xD *m,k* denotes the *m*-th row and *k*-th column entry of X^D, while yD *k,n* denotes the *k*-row and *n*-th column of Y^D. bD *m* and bD *n* denotes the bias of *m*-th instance and *n*-th items. Also, (5.12) can be extended on each single invocation $h_{m,n}$:

$$\text{On } h_{m,n}:\begin{cases} \Delta_{m,n}^D \geq 0 : \varepsilon_{m,n}(X^D,\ Y^D) = \\ \frac{1}{2}\left[\alpha_1^I \Delta_{m,n}^D + \alpha_2\left(\Delta_{m,n}^D\right)^2 + \lambda\left(\sum_{k=1}^{d}\left(\left(x_{m,k}^D\right)^2 + \left(y_{k,u}^D\right)^2\right) + \left(b_m^D\right)^2 + \left(b_n^D\right)^2\right)\right] \\ \Delta_{m,n}^D < 0 : \varepsilon_{m,n}(X^D,\ Y^D) = \\ \frac{1}{2}\left[-\alpha_1^I \Delta_{m,n}^D + \alpha_2\left(\Delta_{m,n}^D\right)^2 + \lambda\left(\sum_{k=1}^{d}\left(\left(x_{m,k}^D\right)^2 + \left(y_{k,n}^D\right)^2\right) + \left(b_m^D\right)^2 + \left(b_n^D\right)^2\right)\right] \end{cases}, \tag{5.13}$$

where $\Delta_{m,n}^D = h_{m,n} - \sqrt{\sum_{k=1}^{d}\left(x_{m,k}^D - y_{k,n}^D\right)^2} - b_m^D - b_n^D - u$. Then minimized (5.13) for the desired X^D and Y^D by SGD:

On $h_{m,n}, \forall k \in \{1, 2, \ldots, d\}$:

$$
\begin{cases}
\Delta^D_{m,n} \geq 0: \begin{cases}
x^D_{m,k} \leftarrow x^D_{m,k} + \frac{1}{2}\alpha_1\eta\left(x^D_{m,k} - y^D_{n,k}\right) / \sqrt{\sum_{k=1}^{d}\left(x^D_{m,k} - y^D_{k,n}\right)^2} \\
+\alpha_2\eta\Delta^D_{m,n}\left(x^D_{m,k} - y^D_{n,k}\right) / \sqrt{\sum_{k=1}^{d}\left(x^D_{m,k} - y^D_{k,n}\right)^2} - \eta\lambda x^D_{m,k} \\
y^D_{n,k} \leftarrow y^D_{n,k} - \frac{1}{2}\alpha_1\eta\left(x^D_{m,k} - y^D_{n,k}\right) / \sqrt{\sum_{k=1}^{d}\left(x^D_{m,k} - y^D_{k,n}\right)^2} \\
-\alpha_2\eta\Delta^D_{m,n}\left(x^D_{m,k} - y^D_{m,k}\right) / \sqrt{\sum_{k=1}^{d}\left(x^D_{m,k} - y^D_{k,n}\right)^2} - \eta\lambda y^D_{m,k} \\
b^D_m \leftarrow b^D_n + \frac{1}{2}\alpha_1\eta + \alpha_2\eta\Delta^D_{m,n} - \eta\lambda b^D_m \\
b^D_n \leftarrow b^D_n + \frac{1}{2}\alpha_1\eta + \alpha_2\eta\Delta^D_{m,n} - \eta\lambda b^D_n
\end{cases} \\
\Delta^D_{m,n} < 0: \begin{cases}
x^D_{m,k} \leftarrow x^D_{m,k} - \frac{1}{2}\alpha_1\eta\left(x^D_{m,k} - y^D_{n,k}\right) / \sqrt{\sum_{k=1}^{d}\left(x^D_{m,k} - y^D_{k,n}\right)^2} \\
+\alpha_2\eta\Delta^D_{m,n}\left(x^D_{m,k} - y^D_{n,k}\right) / \sqrt{\sum_{k=1}^{d}\left(x^D_{m,k} - y^D_{k,n}\right)^2} - \eta\lambda x^D_{m,k} \\
y^D_{n,k} \leftarrow y^D_{n,k} + \frac{1}{2}\alpha_1\eta\left(x^D_{m,k} - y^D_{n,k}\right) / \sqrt{\sum_{k=1}^{d}\left(x^D_{m,k} - y^D_{k,n}\right)^2} \\
-\alpha_2\eta\Delta^D_{m,s}\left(x^D_{m,k} - y^D_{n,k}\right) / \sqrt{\sum_{k=1}^{d}\left(x^D_{m,k} - y^D_{k,n}\right)^2} - \eta\lambda y^D_{n,k} \\
b^D_m \leftarrow b^D_m - \frac{1}{2}\alpha_1\eta + \alpha_2\eta\Delta^D_{m,s} - \eta\lambda b^D_u \\
b^D_n \leftarrow b^D_n - \frac{1}{2}\alpha_1\eta + \alpha_2\eta\Delta^D_{m,n} - \eta\lambda b^D_n
\end{cases}
\end{cases}
\tag{5.14}
$$

αD 1 and αD 2can be self-adaptive according to the training errors of L_1-*norm*- and L_2-*norm*-oriented *Losses*. So their setting formulas are given as follows:

$$L_1^D(i) = \sum_{h_{m,n} \in H_O} |\Delta_{m,n}^D(i)|, L_2^D(i) = \sum_{h_{m,n} \in H_O} \left(\Delta_{m,n}^D(i)\right)^2, \tag{5.15}$$

$$C_{L_1}^D(i) = \sum_{j=1}^{n} L_1^D(j), C_{L_2}^D(i) = \sum_{j=1}^{n} L_2^D(j), \tag{5.16}$$

$$\begin{aligned} \alpha_1^D(i) &= \frac{e^{-\gamma^D(i) C_{L_1}^D(i-1)}}{e^{-\gamma^D(i) C_{L_1}^D(i-1)} + e^{-\gamma^D(i) C_{L_2}^D(i-1)}}, \\ \alpha_2^D(i) &= \frac{e^{-\gamma^D(n) C_{L_2}^D(i-1)}}{e^{-\gamma^D(i) C_{L_1}^D(i-1)} + e^{-\gamma^D(i) C_{L_2}^D(i-1)}}, \end{aligned} \tag{5.17}$$

$$\gamma^D = \frac{1}{C_{L_1}^D(i-1) + C_{L_2}^D(i-1)}, \tag{5.18}$$

where *LD* 1(n) and *LD* 2(n) denote the partial *Loss* of L_1-*norm*- and L_2-*norm*-oriented *Losses*, ΔD *m,n*(i) denotes the state of ΔD *m,n*, the cumulative Loss of *LD* 1(i) and *LD* 2(i) represented by *CD L*1(i) and *CD L*2(i), the state of αD *1 and* αD *2*denoted by αD 1(i) and αD 2(i), and $\gamma^D(i)$ is the balance coefficient. All of them are setting at i-th training iteration on Euclidean distance space.

Based on the above formulas, the HDI predictor can be trained with L_1-and-L_2-*norm*-oriented *Loss* on Euclidean distance space, called D^{2}E-LF-2.

5.3.3 Ensemble of D^{2}E-LF-1 and D^{2}E-LF-2

In machine learning, for several models that are not ideal, and each has its advantages, the ensemble is a great method to combine their advantages [44]. Effective ensemble requires diversification and high accuracy of basic models. D^{2}E-LF-1 and D^{2}E-LF-2, which are based on inner produce space and distance space, respectively, can meet the two requirements of the ensemble. Employing the weighting strategy to ensemble these two predictors to obtain the final result $\hat{H}_{m,n}$ by the following formula:

$$\begin{aligned} \widehat{h}_{m,n} = w &\left(\sum_{k=1}^{d} x_{m,k}^I y_{k,n}^I + b_m^I + b_n^I + u \right) + (1-w) \\ &\times \left(\sqrt{\sum_{k=1}^{d} \left(x_{m,k}^D - y_{k,n}^D \right)^2} + b_m^D + b_n^D + u \right) \end{aligned} \tag{5.19}$$

where w and $(1 - w)$ respectively denote the weights of model D^{2}E-LF-1 and D^{2}E-LF-2. And then, let w self-adaptive according to the prediction errors of two

models, such as αI 1 and αI 2 or αD 1 and αD 2. The prediction errors $E^I(i)$ and $E^D(i)$ at the i-th training iteration can be formulated as follows:

$$E^I(i) = \sum_{h_{m,n} \in H_O} |\Delta^I_{m,n}(i)|, E^D(i) = \sum_{h_{m,n} \in H_K} |\Delta^D_{m,n}(i)|. \tag{5.20}$$

Then, the sum errors $SI\ E(i)$ and $SD\ E(i)$ until the i-th training iteration can be calculated by:

$$S^I_E(i) = \sum_{j=1}^{n} E^I(j), S^D_E(i) = \sum_{j=1}^{n} E^D(j). \tag{5.21}$$

Based on (5.21), at i-th training iteration, $w(i)$ can be self-adjusted with balance coefficient $\gamma^E(i)$ by:

$$w(i) = \frac{e^{-\gamma^E(i) S^I_E(i-1)}}{e^{-\gamma^E(i) S^I_E(i-1)} + e^{-\gamma^E(i) S^D_E(i-1)}} \tag{5.22}$$

During the training processes, $\gamma^E(i)$ is used to control the ensemble of D^{2}E-LF-1 and D^{2}E-LF-2, and can be set adaptively by the following formula:

$$\gamma^E(i) = \frac{1}{S^I_E(i-1) + S^D_E(i-1)} \tag{5.23}$$

5.3.4 Algorithm Design and Analysis

The pseudo-code of D^{2}E-LF is recorded in Table 5.1. In industrial applications, $(|M| + |N|) \gg |H_K|$, so the most time cost of D^{2}E-LF is to train X^I, Y^I, X^D, and Y^D. Therefore, the time complexity of D^{2}E-LF is $O(Iter \times |H_O| \times d)$, where $Iter$ is the iteration number.

5.4 Performance Analysis

5.4.1 General Settings

Datasets *Response Time* and *Throughput* are adopted as two benchmarks HDI datasets in experiments. Response Time and Throughput have 1,873,838 and 1,831,253 records, respectively, generated by 339 users on 5825 different services. The detail of the divided training set and testing set are shown in Table 5.2.

Table 5.1 Algorithm D^{2}E-LF

Steps	Input: H_O
1	**initialize** f, λ, η, αI 1=αI 2=αD 1=αD 2=w=0.5, $Iter$ = 1000
2	**initialize** X^I, Y^I, X^D, and Y^D randomly
3	**for** m=1 to $\|M\|$
4	**initialize** $bI\ m$ and $bD\ m$ randomly
5	**end for**
6	**for** n=1 to $\|N\|$
7	**initialize** $bI\ n$ and $bD\ n$ randomly
8	**end for**
9	**while** $i \leq Iter$ && not converge
10	**for** each rating $h_{m,n}$ in H_O
11	**for** k=1 to d
12	**update** $xI\ m,k$, $yI\ k,n$, $bI\ m$, and $bI\ n$ according to (5.7)
13	**update** $xD\ m,k$, $yD\ k,n$, $bD\ m$, and $bD\ n$ according to (5.14)
14	**end for**
15	**end for**
16	**update** $\alpha I\ 1(i)$ and $\alpha I\ 2(i)$ according to (5.8)–(5.11)
17	**update** $\alpha D\ 1(i)$ and $\alpha D\ 2(i)$ according to (5.15)–(5.18)
18	**update** w according to (5.20)–(5.23)
19	**for** each rating $h_{m,n}$ in H_O
20	**predict** $h_{m,n}$ to obtain $\hat{H}_{m,n}$ according to (5.19)
21	**end for**
22	$i = i$+1
23	**end while**

Table 5.2 The properties of all the divided datasets

Dataset	No.	Density[a] (%)	Training set	Testing set
Response time	D1.1	1	18,738	1,855,100
	D1.2	5	93,692	1,780,146
	D1.3	10	187,384	1,686,454
	D1.4	15	374,768	1,499,070
	D1.5	20	749,535	1,124,303
Throughput	D2.1	1	18,313	1,812,940
	D2.2	5	91,563	1,739,690
	D2.3	10	183,125	1,739,690
	D2.4	15	274,689	1,648,128
	D2.5	20	366,251	1,465,002

[a]Density denotes the percentage of training data compared to all the known records

Baselines D^{2}E-LF is compared with nine relate state-of-the-art-predictors. BLF, RSNMF [44], NIMF [54], NAMF [12], GeoMF [1], LMF-PP [4], DCALF [13], LBFM [58] and AutoRec. To evaluate prediction accuracy, mean absolute error (MAE) and root mean squared error (RMSE) are computed.

5.4.2 Performance Comparison

Comparison of Prediction Accuracy Table 5.3 records the comparison results. In all testing cases, it can be found that D2E-LF has a lower RMSE/MAE, i.e., higher prediction accuracy than the other comparison models.

Comparison of Computational Efficiency The running time on CPU of all the involved models is shown in Fig. 5.4. AutoRec consumes the most CPU running time as it is a deep-learning-based model that needs the full data of HDI matrix to train. And D^2E-LF is ranked second, the reason is that it trains two predictors for ensemble. But D^2E-LF's time complexity $O(N \times |M_K| \times d)$ is linear with $|H_O|$, so in real industry applications, D^2E-LF has suitable computational efficiency.

Influence of Inner Produce Space and Distance Space The weight w increases from 0 to 1 to test the influence of inner produce space and distance space. Figure 5.5 and 5.6 depict the verification results on D1 and D2. The lowest RMSE/MAE is signed by a red circle. And it can find that the results of the ensemble achieve lower RMSE/MAE than D^2E-LF-1 and D^2E-LF-2 applied separately.

5.5 Summary

This chapter introduces a Double-space and Double-norm Ensembled Latent Feature (D^2E-LF) model. Based on the above theoretical analysis and experimental results, it can conclude that D^2E-LF can ensemble multiple advantages from inner product space and distance space, L_1-norm and L_2-norm, to achieve high accuracy on HDI matrix prediction. The main idea of D^2E-LF model is threefold: (1) Double-space. Employed distance space and inner product space to model two kinds of HDI data predictors. (2) Double-norm. Both of these two predictors use an L_1-and-L_2-norm-oriented Loss function. (3) Ensembled. These two predictors are combined by a weighting strategy. In the future, it may further enhance D^2E-LF's performance by: (1) incorporating more metrics into its *Loss*, e.g., $L_{2,1}$-*norm*, and (2)incorporating side and geographical information into its objective function as a regularization constraint.

Table 5.3 The comparison results on prediction accuracy, including win/loss counts and Friedman test

Datasets	Metric	BLFA	RSNMF	NIMF	NAMF	GeoMF	LMF-PP	DCALF	LBFM	AutoRec	D^2E-LF
D1.1	RMSE	1.7671●	1.7529●	1.7512●	1.7446●	1.5824●	1.6024●	1.6649●	1.6272●	1.7461●	1.5376
	MAE	0.7130●	0.6921●	0.6890●	0.6894 ●	0.6781●	0.6632●	0.6530●	0.6124	0.7064●	0.6277
D1.2	RMSE	1.4058●	1.4032●	1.4023●	1.3995●	1.3152●	1.3410●	1.3731●	1.3468●	1.3730●	1.3056
	MAE	0.5561●	0.5438●	0.5502●	0.5465●	0.5305●	0.5285●	0.5127●	0.4558	0.5467●	0.4738
D1.3	RMSE	1.2657●	1.2689●	1.2698●	1.2694●	1.2190●	1.2419●	1.2450●	1.2741●	1.2678●	1.1994
	MAE	0.4944●	0.4868●	0.4842●	0.4976●	0.4827●	0.4725●	0.4544●	0.3792	0.5055●	0.4069
D1.4	RMSE	1.2155●	1.2067●	1.1906●	1.2178●	1.1742●	1.2102●	1.2001●	1.2463●	1.1923●	1.1493
	MAE	0.4691●	0.4492●	0.4508●	0.4625●	0.4495●	0.4472●	0.4346●	0.3614	0.4598●	0.3820
D1.5	RMSE	1.1885●	1.1568●	1.1649●	1.1592●	1.1528●	1.1612●	1.1759●	1.2014●	1.1681●	1.1256
	MAE	0.4531●	0.4371●	0.4346●	0.4360●	0.4366●	0.4260●	0.4246●	0.3478	0.4482●	0.3687
D2.1	RMSE	84.7661●	84.5423●	84.1428●	85.7821●	88.6521●	85.1048●	83.2891●	86.2188●	85.1214●	80.4369
	MAE	32.3966	32.9940	31.8251	34.8227●	37.2881●	31.9704	31.9826	33.4118●	32.5432	33.2675
D2.2	RMSE	51.6982●	60.7994●	51.4585●	53.9572●	57.7842●	51.7765●	51.4123●	54.6529●	55.5352●	50.9875
	MAE	18.9776●	21.4302●	17.7153	20.2104●	24.7465●	18.3091	18.62373●	19.6723●	21.3118●	18.5502
D2.3	RMSE	46.5606●	50.5298●	45.8432●	46.0215●	49.2456●	46.1418●	45.9013●	45.9237●	48.4771●	45.4089
	MAE	16.1924●	17.2305●	15.5264	17.0126●	22.4728●	15.9125	15.3430	15.8224	17.0310●	16.1166
D2.4	RMSE	43.6210●	45.2647●	43.0232●	43.6522●	45.3255●	42.9927●	42.6235●	42.4228●	44.5246●	41.9991
	MAE	14.9279●	14.6880	14.2146	15.6547●	17.7908●	14.7450●	14.0664	14.3665	15.0156●	14.7267
D2.5	RMSE	41.7846●	43.5882●	41.0765●	41.3523●	43.9845●	41.4084●	41.2194●	40.5023●	43.0654●	40.3075
	MAE	14.3061●	14.3654●	13.5638	14.6482●	16.2852●	14.1033●	13.5491	12.5482	14.2265●	14.0116
Statistic	Win/loss	19/1	18/2	15/5	20/0	20/0	17/3	16/4	12/8	19/1	**156/24**[a]
	F-rank*	7.4	7.3	4.6	7.2	6.8	4.35	3.4	4.4	7.35	**2.2**

[a] The total win/loss cases of D^2E-LF. * A lower F-rank value indicates a higher prediction accuracy. Where ● denotes that D^2E-LF has a lower RMSE/MAE than comparison models

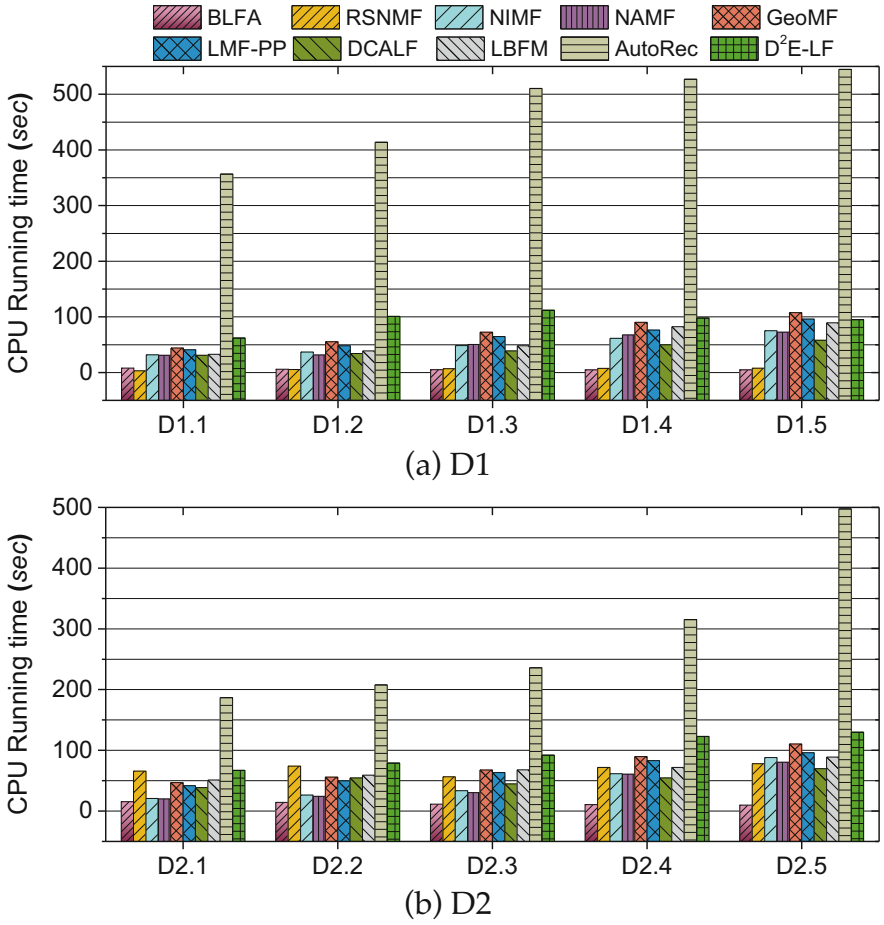


Fig. 5.4 CPU running time of comparison models. (a) D1, (b) D2

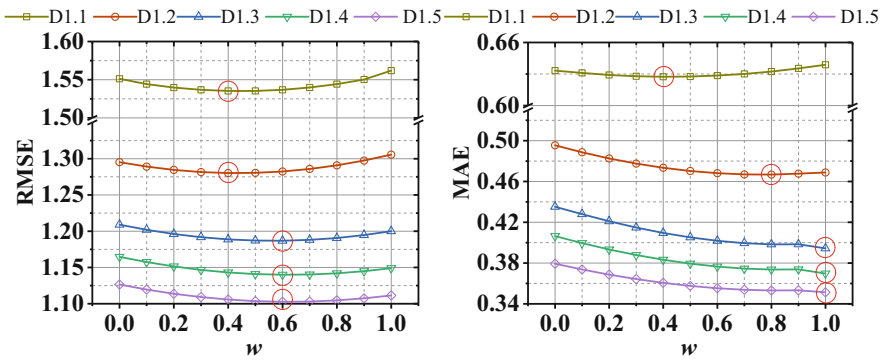


Fig. 5.5 The RMSE and MAE of D^{2}E-LF on D1 when w increasing from 0 to 1

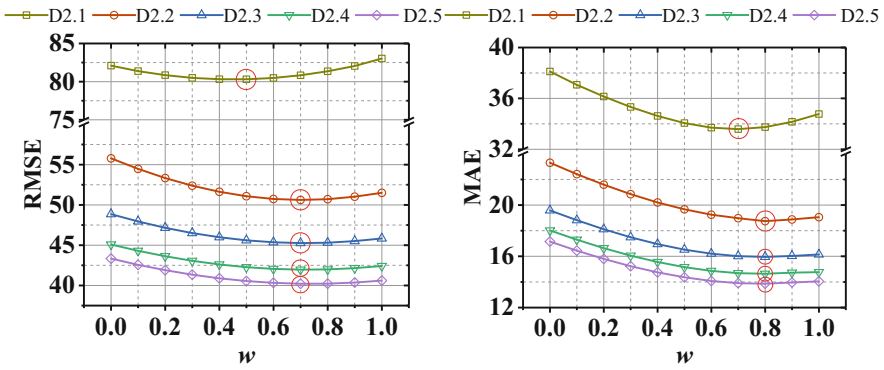


Fig. 5.6 The RMSE and MAE of D^{2}E-LF on D2 when w increasing from 0 to 1

References

1. Chen, Z., Shen, L., Li, F., You, D.: Your neighbors alleviate cold-start: on geographical neighborhood influence to collaborative web service QoS prediction. Knowl.-Based Syst. **138**, 188–201 (2017)
2. Wu, D., Luo, X., Wang, G., Shang, M., Yuan, Y., Yan, H.: A highly accurate framework for self-labeled semisupervised classification in industrial applications. IEEE Trans. Ind. Inform. **14**(3), 909–920 (2018)
3. Deng, S., Zhang, J., Wu, D., He, Y., Xie, X., Wu, X.: A quantitative risk assessment model for distribution cyber physical system under cyber attack. In: IEEE Transactions on Industrial Informatics, p. 1 (2022)
4. Ryu, D., Lee, K., Baik, J.: Location-based web service QoS prediction via preference propagation to address cold start problem. IEEE Trans. Serv. Comput. **14**(3), 736–746 (2021). https://doi.org/10.1109/TSC.2018.2821686
5. Chang, J., Guo, B., Yao, Q.: High dimensional stochastic regression with latent factors, endogeneity and nonlinearity. J. Econ. **189**(2), 297–312 (2015)
6. He, Y., Wu, B., Wu, D., Beyazit, E., Chen, S., Wu, X.: Toward mining capricious data streams: a generative approach. IEEE Trans. Neural Netw. Learn. Syst. **32**(3), 1228–1240 (2021)
7. Shang, M., Yuan, Y., Luo, X., Zhou, M.: An α–β-divergence-generalized recommender for highly accurate predictions of missing user preferences. IEEE Trans. Cybern. **52**(8), 8006–8018 (2022). https://doi.org/10.1109/TCYB.2020.3026425
8. Wu, H., Luo, X., Zhou, M.: Advancing non-negative latent factorization of tensors with diversified regularization schemes. IEEE Trans. Serv. Comput. **15**(3), 1334–1344 (2022). https://doi.org/10.1109/TSC.2020.2988760
9. Wu, D., He, Y., Luo, X., Zhou, M.: A latent factor analysis-based approach to online sparse streaming feature selection. IEEE Trans. Syst. Man Cybern. Syst. **52**, 6744–6758 (2021)
10. Yuan, Y., He, Q., Luo, X., Shang, M.: A multilayered-and-randomized latent factor model for high-dimensional and sparse matrices. IEEE Trans. Big Data. **8**(3), 784–794 (2022). https://doi.org/10.1109/TBDATA.2020.2988778
11. Shi, X., He, Q., Luo, X., Bai, Y., Shang, M.: Large-scale and scalable latent factor analysis via distributed alternative stochastic gradient descent for recommender systems. IEEE Trans. Big Data. **8**(2), 420–431 (2022). https://doi.org/10.1109/TBDATA.2020.2973141
12. Tang, M., Zheng, Z., Kang, G., Liu, J., Yang, Y., Zhang, T.: Collaborative web service quality prediction via exploiting matrix factorization and network map. IEEE Trans. Netw. Serv. Manag. **13**(1), 126–137 (2016). https://doi.org/10.1109/TNSM.2016.2517097

13. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Wu, X.: A data-characteristic-aware latent factor model for web services QoS prediction. IEEE Trans. Knowl. Data Eng. **34**(6), 2525–2538 (2022). https://doi.org/10.1109/TKDE.2020.3014302
14. Wu, D., He, Q., Luo, X., Shang, M., He, Y., Wang, G.: A posterior-neighborhood-regularized latent factor model for highly accurate web service QoS prediction. IEEE Trans. Serv. Comput. **15**(2), 793–805 (2022). https://doi.org/10.1109/TSC.2019.2961895
15. Wu, D., Shang, M., Luo, X., Wang, Z.: An L_1-and-L_2-norm-oriented latent factor model for recommender systems. IEEE Trans. Neural Netw. Learn. Syst. **33**, 1–14 (2021). https://doi.org/10.1109/TNNLS.2021.3071392
16. Lucas, J.E., Kung, H.-N., Chi, J.-T.A.: Latent factor analysis to discover pathway-associated putative segmental aneuploidies in human cancers. PLoS Comput. Biol. **6**(9), e1000920 (2010)
17. Waris, O., et al.: A latent factor analysis of working memory measures using large-scale data. Front. Psychol. **8**, 1062 (2017)
18. Luo, X., Yuan, Y., Wu, D.: Adaptive regularization-incorporated latent factor analysis. In: 2020 IEEE International Conference on Knowledge Graph (ICKG), pp. 481–488 (2020)
19. Luo, X., Shang, M., Li, S.: Efficient extraction of non-negative latent factors from high-dimensional and sparse matrices in industrial applications. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 311–319 (2016). https://doi.org/10.1109/ICDM.2016.0042
20. Sun, B., Wu, D., Shang, M., He, Y.: Toward auto-learning hyperparameters for deep learning-based recommender systems. In: Database Systems for Advanced Applications, pp. 323–331. Springer, Cham (2022)
21. Luo, X., Zhou, Y., Liu, Z., Hu, L., Zhou, M.: Generalized Nesterov's acceleration-incorporated non-negative and adaptive latent factor analysis. IEEE Trans. Serv. Comput. **15**, 1–2823 (2021). https://doi.org/10.1109/TSC.2021.3069108
22. Wu, D., Zhang, P., He, Y., Luo, X.: A double-space and double-norm ensembled latent factor model for highly accurate web service QoS prediction. IEEE Trans. Serv. Comput., 1 (2022). https://doi.org/10.1109/TSC.2022.3178543
23. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Zhou, M.: A deep latent factor model for high-dimensional and sparse matrices in recommender systems. IEEE Trans. Syst. Man Cybern. Syst. **51**(7), 4285–4296 (2021). https://doi.org/10.1109/TSMC.2019.2931393
24. Wu, D., Luo, X., He, Y., Zhou, M.: A prediction-sampling-based multilayer-structured latent factor model for accurate representation to high-dimensional and sparse data. In: EEE Transactions on Neural Networks and Learning Systems, pp. 1–14 (2022)
25. Jiang, J., Li, W., Dong, A., Gou, Q., Luo, X.: A fast deep AutoEncoder for high-dimensional and sparse matrices in recommender systems. Neurocomputing. **412**, 381–391 (2020)
26. Wang, D., et al.: Elastic-net regularized latent factor analysis-based models for recommender systems. Neurocomputing. **329**, 66–74 (2019)
27. Shi, X., Luo, X., Shang, M., Gu, L.: Long-term performance of collaborative filtering based recommenders in temporally evolving systems. Neurocomputing. **267**, 635–643 (2017)
28. Zhang, S., Yao, L., Wu, B., Xu, X., Zhang, X., Zhu, L.: Unraveling metric vector spaces with factorization for recommendation. IEEE Trans. Ind. Inform. **16**(2), 732–742 (2020). https://doi.org/10.1109/TII.2019.2947112
29. Hsieh, C.-K., Yang, L., Cui, Y., Lin, T.-Y., Belongie, S., Estrin, D.: Collaborative metric learning. In: Proceedings of the 26th International Conference on World Wide Web, Perth Australia, pp. 193–201 (2017). https://doi.org/10.1145/3038912.3052639
30. Zhu, X., et al.: Similarity-maintaining privacy preservation and location-aware low-rank matrix factorization for QoS prediction based web service recommendation. IEEE Trans. Serv. Comput. **14**(3), 889–902 (2021). https://doi.org/10.1109/TSC.2018.2839741
31. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. arXiv preprint arXiv:1301.7363 (2013)

32. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Wu, X.: A data-aware latent factor model for web service QoS prediction. In: Advances in Knowledge Discovery and Data Mining, pp. 384–399. Springer, Cham (2019)
33. Luo, X., Zhou, M., Xia, Y., Zhu, Q.: An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. IEEE Trans. Ind. Inform. **10**(2), 1273–1284 (2014)
34. Luo, X., Xia, Y., Zhu, Q.: Applying the learning rate adaptation to the matrix factorization based collaborative filtering. Knowl.-Based Syst. **37**, 154–164 (2013)
35. Wu, H., Yue, K., Li, B., Zhang, B., Hsu, C.-H.: Collaborative QoS prediction with context-sensitive matrix factorization. Futur. Gener. Comput. Syst. **82**, 669–678 (2018)
36. Luo, X., Xia, Y., Zhu, Q.: Incremental collaborative filtering recommender based on regularized matrix factorization. Knowl.-Based Syst. **27**, 271–280 (2012)
37. Luo, X., Liu, Z., Jin, L., Zhou, Y., Zhou, M.: Symmetric nonnegative matrix factorization-based community detection models and their convergence analysis. IEEE Trans. Neural Netw. Learn. Syst. **33**, 1203 (2021)
38. Luo, X., Liu, Z., Li, S., Shang, M., Wang, Z.: A fast non-negative latent factor model based on generalized momentum method. IEEE Trans. Syst. Man Cybern. Syst. **51**(1), 610–620 (2018)
39. Luo, X., Wu, H., Yuan, H., Zhou, M.: Temporal pattern-aware QoS prediction via biased non-negative latent factorization of tensors. IEEE Trans. Cybern. **50**(5), 1798–1809 (2020). https://doi.org/10.1109/TCYB.2019.2903736
40. Luo, X., et al.: Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data. IEEE Trans. Cybern. **48**(4), 1216–1228 (2018). https://doi.org/10.1109/TCYB.2017.2685521
41. Xin, L., Yuan, Y., Zhou, M., Liu, Z., Shang, M.: Non-negative latent factor model based on β-divergence for recommender systems. IEEE Trans. Syst. Man Cybern. Syst. **51**(8), 4612–4623 (2021). https://doi.org/10.1109/TSMC.2019.2931468
42. Hu, L., Yuan, X., Liu, X., Xiong, S., Luo, X.: Efficiently detecting protein complexes from protein interaction networks via alternating direction method of multipliers. IEEE/ACM Trans. Comput. Biol. Bioinform. **16**(6), 1922–1935 (2019). https://doi.org/10.1109/TCBB.2018.2844256
43. Luo, X., Wang, D., Zhou, M., Yuan, H.: Latent factor-based recommenders relying on extended stochastic gradient descent algorithms. IEEE Trans. Syst. Man Cybern. Syst. **51**(2), 916–926 (2021). https://doi.org/10.1109/TSMC.2018.2884191
44. Luo, X., Zhou, M., Xia, Y., Zhu, Q., Ammari, A.C., Alabdulwahab, A.: Generating highly accurate predictions for missing QoS data via aggregating nonnegative latent factor models. IEEE Trans. Neural Netw. Learn. Syst. **27**(3), 524–537 (2016). https://doi.org/10.1109/TNNLS.2015.2412037
45. Wu, D., Jin, L., Luo, X.: PMLF: prediction-sampling-based multilayer-structured latent factor analysis. In: 2020 IEEE International Conference on Data Mining (ICDM), pp. 671–680 (2020)
46. Luo, X., Li, S., Zhou, M.: Regularizaed extraction of non-negative latent factors from high-dimensional sparse matrices. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 001221–001226 (2016)
47. Liu, Z., Luo, X., Wang, Z.: Convergence analysis of single latent factor-dependent, nonnegative, and multiplicative update-based nonnegative latent factor models. IEEE Trans. Neural Netw. Learn. Syst. **32**(4), 1737–1749 (2020)
48. Luo, X., Wang, Z., Shang, M.: An instance-frequency-weighted regularization scheme for non-negative latent factor analysis on high-dimensional and sparse data. IEEE Trans. Syst. Man Cybern. Syst. **51**(6), 3522–3532 (2019)
49. Shang, M., Luo, X., Liu, Z., Chen, J., Yuan, Y., Zhou, M.: Randomized latent factor model for high-dimensional and sparse matrices from industrial applications. IEEE/CAA J. Autom. Sin. **6**(1), 131–141 (2018)

50. Luo, X., Zhou, M., Wang, Z., Xia, Y., Zhu, Q.: An effective scheme for QoS estimation via alternating direction method-based matrix factorization. IEEE Trans. Serv. Comput. **12**(4), 503–518 (2016)
51. Wu, D., et al.: Self-training semi-supervised classification based on density peaks of data. Neurocomputing. **275**, 180–191 (2018)
52. Luo, X., Ouyang, Y., Xiong, Z.: Improving neighborhood based collaborative filtering via integrated folksonomy information. Pattern Recogn. Lett. **33**(3), 263–270 (2012)
53. Wu, D., Luo, X., Wang, G., Shang, M., Yuan, Y., Yan, H.: A highly accurate framework for self-labeled semisupervised classification in industrial applications. IEEE Trans. Ind. Inform. **14**(3), 909–920 (2018). https://doi.org/10.1109/TII.2017.2737827
54. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Collaborative web service QoS prediction via neighborhood integrated matrix factorization. IEEE Trans. Serv. Comput. **6**(3), 289–299 (2013). https://doi.org/10.1109/TSC.2011.59
55. Zhu, J., He, P., Zheng, Z., Lyu, M.R.: Online QoS prediction for runtime service adaptation via adaptive matrix factorization. IEEE Trans. Parallel Distrib. Syst. **28**(10), 2911–2924 (2017). https://doi.org/10.1109/TPDS.2017.2700796
56. Luo, Z., Liu, L., Yin, J., Li, Y., Wu, Z.: Latent ability model: a generative probabilistic learning framework for workforce analytics. IEEE Trans. Knowl. Data Eng. **31**(5), 923–937 (2019). https://doi.org/10.1109/TKDE.2018.2848658
57. Zhang, Y., et al.: Covering-based web service quality prediction via neighborhood-aware matrix factorization. IEEE Trans. Serv. Comput. **14**(5), 1333–1344 (2021). https://doi.org/10.1109/TSC.2019.2891517
58. Yang, Y., Zheng, Z., Niu, X., Tang, M., Lu, Y., Liao, X.: A location-based factorization machine model for web service QoS prediction. IEEE Trans. Serv. Comput. **14**(5), 1264–1277 (2021). https://doi.org/10.1109/TSC.2018.2876532

Chapter 6
Data-characteristic-aware Latent Feature Learning

6.1 Overview

Some state-of-the-art missing data predictors are primarily based on a latent feature learning (LFL)-based model [1–14]. They improved the basic LFL model based on the neighborhood information of historical recorded data [15–17]. While they have some limitations as follows:

1. They identify the neighborhood primarily based on the common set defined on the original high dimensional and incomplete (HDI) matrix, which may not be accurate. Give an example of quality of service (QoS) data prediction in Fig. 6.1, after identifying the common sets among users, some observed QoS data (selected in the red entries) would be abandoned, leading to there being no common QoS records behind in the target user u and user b. A similarity in user u and user b cannot be correctly identified on the original sparse user-service QoS matrix even if they have something in common with user a.
2. They use the given HDI data directly while ignoring the reliability of the data. Hence, they cannot handle noise (unreliable HDI data) caused by call failure or accidents [18–20]. Such as the noise data of user c in Fig. 6.1, can seriously affect the performance of the resulting model.

To address the above problems, a data-characteristic-aware latent factor (DCALF) model is proposed in [55]. Its main idea is towfold: (1) it first extracts the dense latent features from the original raw HDI data by an LFL model, and (2) it employs DPClust method [21] to simultaneously identify the neighborhoods and outliers of HDI data on the extracted latent features.

D. Wu, *Robust Latent Feature Learning for Incomplete Big Data*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-19-8140-1_6

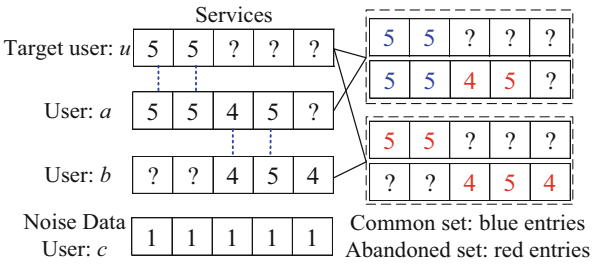


Fig. 6.1 An example of identifying unreliable neighborhoods based on raw QoS data

6.2 Related Work

6.2.1 Related LFL-Based Models

Considering existing collaborative filtering (CF)-based HDI data predictors [22–24], an LFL-based one is very popular and widely used [25–28]. Such as in QoS predictors, Luo et al. [29] incorporated non-negative constraints and regularization into LFL model to develop a QoS predictor. Yu et al. [30] introduced a latent neighbor model into LFL model to obtain highly accurate QoS predictions. Zheng et al. [31] developed an LFL model based on the information of user similarity. According to data transformation and online learning, Zhu et al. [32] realized online QoS prediction by an adaptive LFL model.

In general, when users in the same area, they may have a similar calling environment. So the geographical information is an impact factor in predicting user's actions [30]. It can be used as additional input for an LFL model [33–39]. Kumar and Anouncia [40] utilized this information to improve the accuracy of Web service selection. Chen et al. [15] incorporated geographical neighborhoods into LFL model to achieve highly accurate prediction. Tang et al. [41] proposed a network-aware LFL-based QoS predictor based on the user network map. According to network location-aware neighbor selection, Yin et al. [42] proposed a hybrid QoS predictor. Through systematically modeling geographical information, sample set diversity, and platform context, a QoS predictor based on neighbor-aware LFL-based was proposed by Feng and Huang et al. [5]. Adopted the local information of users and services, an LFL-based QoS predictor was proposed by Ryu et al. [43]. Different from these models, DCALF can simultaneously identify the neighborhoods and outliers of HDI data.

6.2.2 DPClust Algorithm

DPClust is a clustering algorithm proposed in [21]. DPClust characterizes its cluster centers according to data density. Given a dataset $Z = \{z_1, z_2, \ldots, z_G\}$, for each data

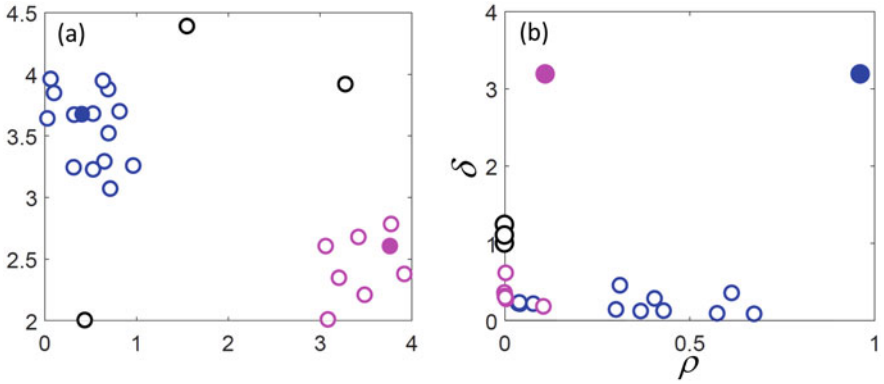


Fig. 6.2 The example of DPClust: (a) data distribution; (b) decision graph for data in (a)

point z_i where $i \in \{1, 2, \ldots, G\}$, let ρ_i denote local density. ρ_i can be computed by a kernel function such as a cut-off as follows:

$$\rho_i = \sum_{j=1, j\neq i}^{N} \Phi\left(dist_{i,j} - dist_c\right), \quad \Phi(t) = \begin{cases} 1 & t < 0 \\ 0, & others \end{cases} \tag{6.1}$$

where the distance $dist_{i,j}$ is between z_i and z_j, and $dist_c$ is the cutoff constant, respectively. On the other hand, with a Gaussian kernel, ρ_i is formulated by:

$$\rho_i = \sum_{j=1, j\neq i}^{G} e^{-\left(\frac{dist_{i,j}}{dist_c}\right)^2}. \tag{6.2}$$

Note that for robustness [21], $dist_c$ is set as follows.

$$Vec = sort\left(d_{ij}\right), dist_c = Vec(\lfloor P_{Vec} \times G \times (G-1)/2 \rfloor), \tag{6.3}$$

when sorting all $dist_{i,j}$ in ascending order, it can obtain a vector Vec. Using the percentage P_{Vec} to denote the average percentage of all data points neighbors. According to [21], P_{Vec} is usually set in the [1%, 2%] interval.

For each z_i, let δ_i denotes the minimum distance between z_i and any other data point with higher local density:

$$\delta_i = \begin{cases} \max_j\left(dist_{ij}\right), & \forall j, \rho_i \geq \rho_j, \\ \min_{j:\rho_i < \rho_j}\left(dist_{i,j}\right), & otherwise. \end{cases} \tag{6.4}$$

Thus, cluster centers are recognized as the data points with anomalously large ρ and δ. To understand DPClust easily, Fig. 6.2 gives a simple example of DPClust.

Figure 6.2(a) shows in a two-dimensional space, 26 data points were embedded. All data points will be computed ρ and δ, the decision graph [21] that is the plot of δ as a function of ρ for each data point, is drawn in Fig. 6.2(b). Then, the DPClust algorithm recognizes the blue solid and the pink solid points as cluster centers. The three black hollow data points are outliers as they have a relatively small ρ and a large δ. Thus, by computing outlier factor γ_i for each z_i, DPClust algorithm is able to detect outliers

$$\gamma_i = \rho_i / \delta_i. \tag{6.5}$$

6.3 A Data-Characteristic-Aware Latent Feature Model

6.3.1 Model Structure

DCALF model consists of three steps, as shown in Fig. 6.3. Step 1: for an HDI matrix **H**, extract latent feature matrices **X**, **Y** for users and services, respectively. Step 2: identify neighborhoods and detect unreliable data with DPClust algorithm.

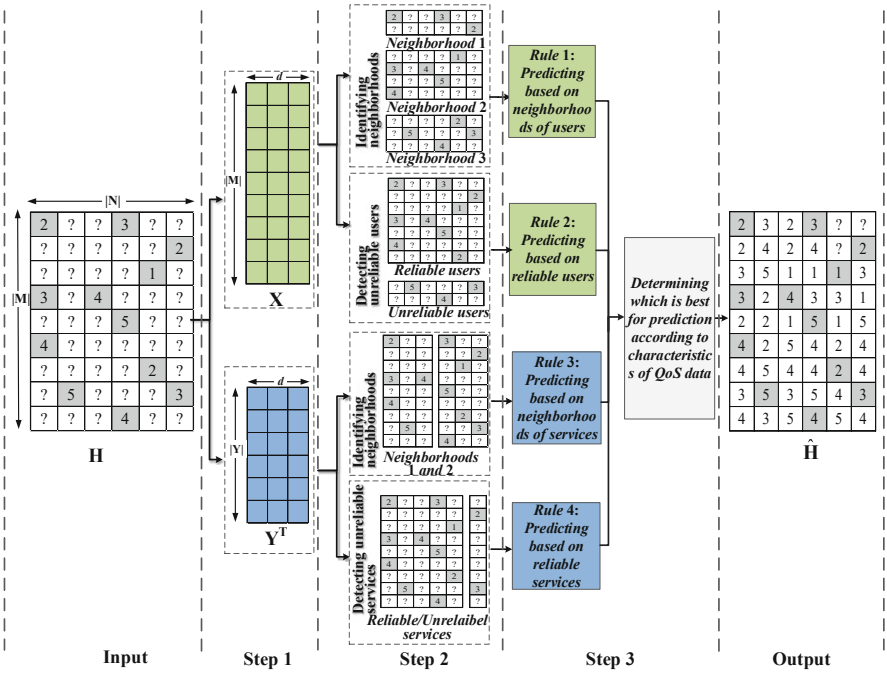


Fig. 6.3 Flowchart of a DCALF model

Step 3: predict the missing data in **H**. Then, these three steps will be described in detail.

6.3.2 Step 1: Latent Feature Extraction

In Step 1, let $\mathbf{H_O}$ and $\mathbf{H_U}$ respectively denote **H**'s known and unknown entry sets. **X** and **Y** are extracted based on $\mathbf{H_O}$ by an LFL model [44–54] as follows:

$$\varepsilon_{m,n} = \frac{1}{2}\left(h_{m,n} - \sum_{j=1}^{d} x_{m,j}y_{j,n}\right)^2 + \frac{\lambda}{2}\left(\sum_{j=1}^{d} x_{m,j}^2 + \sum_{j=1}^{d} y_{j,n}^2\right). \tag{6.6}$$

By using SGD to minimize (6.6), the training rules are as follows:

$$\begin{aligned} &On\ h_{m,n}, for\ k = 1 \sim d: \\ &\begin{cases} x_{m,k} \leftarrow x_{m,k} + \eta y_{k,n}\left(h_{m,n} - \sum\limits_{k=1}^{d} x_{m,k}y_{k,n}\right) - \lambda x_{m,k}, \\ y_{k,n} \leftarrow y_{k,n} + \eta x_{m,k}\left(h_{m,n} - \sum\limits_{k=1}^{d} x_{m,k}y_{k,n}\right) - \lambda y_{k,n}. \end{cases} \end{aligned} \tag{6.7}$$

where λ is the hyper-parameter of learning rate. By repeating (6.7) on $\mathbf{H_O}$ iteratively, **X** and **Y** can be extracted from $\mathbf{H_O}$.

6.3.3 Step 2: Neighborhood and Outlier Detection

X and **Y** can be used to identify neighborhoods and detect unreliable HDI data [55]. The parameter α is used to denote the ratio of unreliable HDI data.

(a) Row neighborhoods and outliers.

With extracted **X**, row m's local density ρ_m is calculated by a cut-off kernel as follows:

$$\rho_m = \sum_{m'=1, m' \neq m}^{|M|} \Phi(dist_{m,m'} - dist_M),\ \Phi(t) = \begin{cases} 1 & t < 0 \\ 0, & otherwise \end{cases} \tag{6.8}$$

where m' represents another row different from row m, and $dist_M$ is the cutoff constant about row set M. Then, the distance $dist_{m,m'}$ between m and m' can be calculated by Euclidean distance.

$$dist_{m,m'} = \sqrt{\sum_{k=1}^{d} (x_{m,k} - x_{m\prime,k})^2}. \tag{6.9}$$

The local density ρ_m also can be computed by Gaussian kernel by the formula:

$$\rho_m = \sum_{m'=1,m'\neq m}^{|M|} e^{-\left(\frac{dist_{m,m\prime}}{dist_M}\right)^2} \tag{6.10}$$

Then, calculate the cutoff constant $dist_M$. According to DPClust algorithm, the $dist_M$ can be computed by formula (6.11).

$$Vec = sort(\ dist_{m,m'}\ ), dist_M = Vec(\lfloor P_{Vec} \times |M| \times (|M| - 1)/2 \rfloor). \tag{6.11}$$

So the minimum distance δ_m of m between m and m' is calculated as follows:

$$\delta_m = \begin{cases} \max_{m'}(dist_{m,m'}), & \forall m', \rho_m \geq \rho_{m'}, \\ \min_{m':\rho_m < \rho_{m'}}(dist_{m,m'}), & otherwise. \end{cases} \tag{6.12}$$

Note that m' has a high local density. Finally, make ρ_m to be divided by δ_m, and the outlier factor γ_m of m is obtained as follows.

$$\gamma_m = \rho_m / \delta_m. \tag{6.13}$$

Then, the original HDI matrix **H** can be cut apart into S small matrices recorded as $\{HM\ 1, HM\ 2, \ldots, HM\ S\}$. Also, separating **H** into two matrices *HM R*and *HM U* to represent the reliable and unreliable ones, respectively.

(b) Column neighborhoods and outliers

For extracted Y from H, the local density ρy of each service y will be calculated by a cutoff kernel and a Gaussian kernel as follows:

$$\rho_y = \sum_{n'=1,n'\neq n}^{|N|} \Phi(dist_{n,n'} - dist_N),\ \Phi(t) = \begin{cases} 1 & t<0 \\ 0, & others \end{cases} \tag{6.14}$$

$$\rho_n = \sum_{n'=1,n'\neq n}^{|N|} e^{-\left(\frac{dist_{n,n\prime}}{dist_N}\right)^2}, \tag{6.15}$$

where n' is another column that is different from n, and distN is the cutoff constant about column set N. Then, the distance distn,n' between n and n' can be calculated by Euclidean distance.

$$dist_{n,n'} = \sqrt{\sum_{k=1}^{d} \left(y_{k,n} - y_{k,n\prime}\right)^2}, \tag{6.16}$$

$$Vec = sort(\ dist_{n,n'}\ ), dist_N = Vec(\lfloor P_{Vec} \times |N| \times (|N| - 1)/2 \rfloor). \tag{6.17}$$

PVec denotes the average percentage of all data points neighbors and is set in the [1%, 2%] interval. So the minimum distance δM between s and s' that have higher local density is calculated as follows:

$$\delta_n = \begin{cases} \max_{n'}(dist_{n,n'}), & \forall n', \rho_n \geq \rho_{n'}, \\ \min_{n':\rho_n < \rho_{n'}}(dist_{n,n'}), & otherwise. \end{cases} \tag{6.18}$$

The outlier factor γn of services is computed by:

$$\gamma_n = \rho_n / \delta_n. \tag{6.19}$$

Similarly, H can be cut apart into S small matrices recorded as {HN 1, HN 2, . . ., HN S}. Also, separating H into two matrices HN Rand HN U to represent the reliable and unreliable ones, respectively.

6.3.4 Step 3: Prediction

According to {*HM* 1, *HM* 2, . . ., *HM S*}, {*HM R*, *HM U*}, {*HN* 1, *HN* 2, . . ., *HN S*}, {*HN R*, *HN U*} which obtained in Steps 1–2 [55], the missing data in **H** can be predicted precisely. Figure 6.4 depicts a flowchart to illustrate the policy of prediction rule for a DCALF model. Figure 6.5 illustrates the four kinds of decision graphs corresponding to different prediction rules. If the user has two or more cluster centers, as shown in Fig. 6.5(a), then they would have two or more neighborhoods. In this case, Prediction Rule 1 would be chosen to make a prediction according to the neighborhoods of users. Also, if there are many outliers, depicted by the red rectangles that are shown in Fig. 6.5(b), the Prediction Rule 2 would be chosen, making a prediction by reliable users. On the other side, the decision process of service is the same as the above-mentioned. If the service has tow or more cluster centers, such as Fig. 6.5(c), the Prediction Rule 3 would be worked on service neighborhoods. In Fig. 6.5(d), if the service has many outliers, the red rectangles, the Prediction Rule 4 would be worked on reliable services.

Using a general function F^{LFL} to denote extracting **X** and **Y** from **H** as follows:

$$\{X, Y\} = F^{LFL}(X, Y|H). \tag{6.20}$$

The four prediction rules could be formulated as follows.

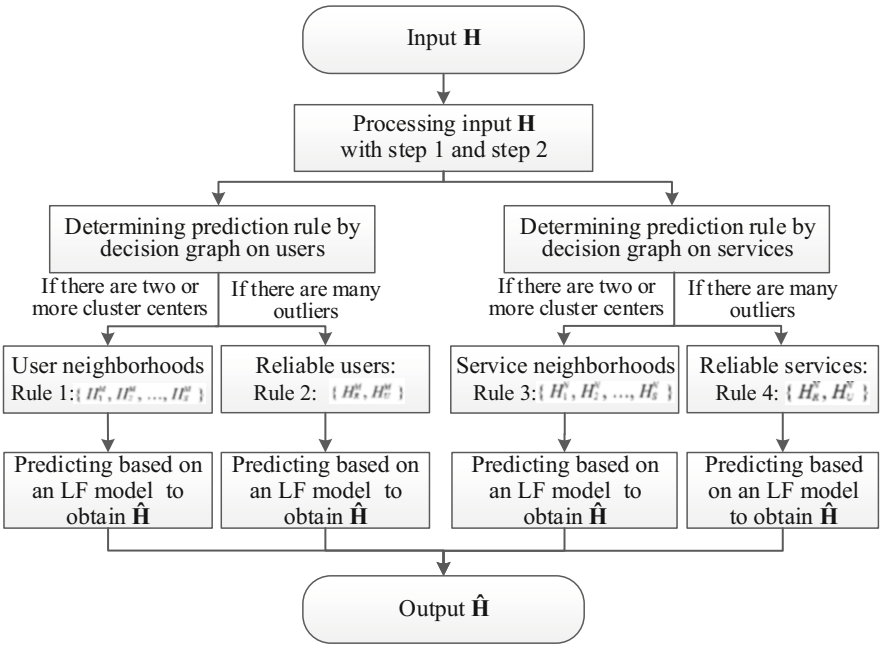


Fig. 6.4 Flowchart of prediction rule selection in a DCALF model

Prediction Rule 1 Based on the neighborhoods of users, it can predict $\hat{\mathbf{H}}$ by the neighbor set {*HM* 1, *HM 2*, . . ., *HM S*}.

$$for\ \mathrm{s}=1\sim S:\ \ \left\{X_s^M,\ \mathrm{Y}_s^M\right\}=F^{LFL}\left(X_s^M,\ Y_s^M|H_s^M\right), \tag{6.21}$$

$$for\ s=1\sim S:\ \ \widehat{\mathrm{H}}_s^M=X_s^M Y_s^M, \tag{6.22}$$

$$\widehat{H}=\widehat{H}_1^M\cup\widehat{H}_2^M\cup\ldots\cup\widehat{H}_s^M. \tag{6.23}$$

Prediction Rule 2 Based on reliable users, $\hat{\mathbf{H}}$ can be obtained by the set {*HM R*, *HM U*} as follows,

$$\left\{X_R^M,\ Y_R^M\right\}=F^{LFL}\left(X_R^M,\ Y_R^M|H_R^M\right), \tag{6.24}$$

$$\widehat{H}_R^M=X_R^M Y_R^M, \tag{6.25}$$

$$\widehat{H}=\widehat{H}_R^M\bigoplus XY|_{row}, \tag{6.26}$$

where $\hat{H}M\ R\bigoplus XY|_{row}$ denotes the corresponding rows of the matrix product of **XY** was replaced by $\hat{H}M\ R$.

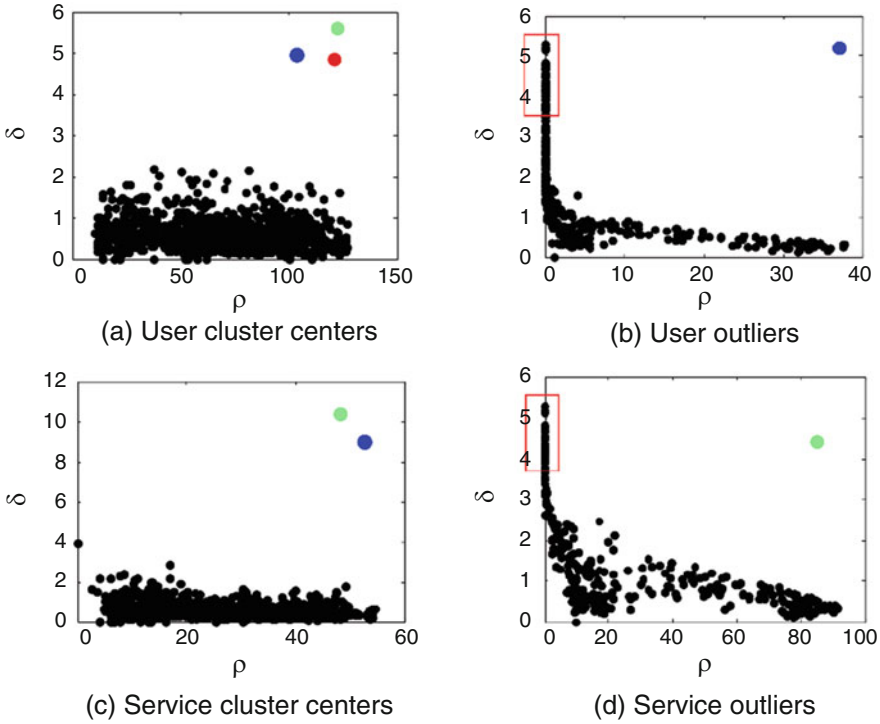


Fig. 6.5 Illustration of four kinds of decision graphs corresponding to different prediction rules

Prediction Rule 3 Based on neighborhoods of services, $\hat{\mathbf{H}}$ can be obtained by the set $\{HN\ 1, HN\ 2, \ldots, HN\ S\}$,

$$for\ \mathrm{s} = 1 \sim S: \quad \left\{X_s^N, \mathrm{Y}_s^N\right\} = F^{LFL}\left(X_s^N, Y_s^N | H_s^N\right), \tag{6.27}$$

$$for\ s = 1 \sim S: \quad \widehat{\mathrm{H}}_s^N = X_s^N Y_s^N, \tag{6.28}$$

$$\widehat{H} = \widehat{H}_1^N \cup \widehat{H}_2^N \cup \ldots \cup \widehat{H}_s^N. \tag{6.29}$$

Prediction Rule 4 Based on reliable services, $\hat{\mathbf{H}}$ can be obtained by the set$\{HN\ R, HN\ U\}$ in the following form,

$$\left\{X_R^N, Y_R^N\right\} = F^{LFL}\left(X_R^N, Y_R^N | H_R^N\right), \tag{6.30}$$

$$\widehat{H}_R^N = X_R^N Y_R^N, \tag{6.31}$$

$$\widehat{H} = \widehat{H}_R^N \bigoplus XY|_{column}, \tag{6.32}$$

And $\hat{HN}\ R \bigoplus XY|_{column}$ denotes the corresponding columns of the matrix product of **XY** was replaced by $\hat{HM}\ R$.

6.4 Performance Analysis

Datasets *Response Time* and *Throughput* are adopted as two benchmarks HDI datasets in experiments [41]. Response Time and Throughput have 1,873,838 and 1,831,253 records, respectively, generated by 339 users on 5825 different services. The detail of the divided training set and testing set are shown in Table 6.1.

Baselines Eight state-of-the-art predictors are used to compare with DCALF regarding the prediction accuracy and computational efficiency. They are BLF, RSNMF [29], LN-LFM [30], NIME [31], NAME [41], GeoMF [15], LMF-PP [43], AutoRec. To evaluate prediction accuracy, mean absolute error (MAE) and root mean squared error (RMSE) are computed.

6.4.1 Prediction Rule Selection

For given D1.4 and D2.4, this section introduces how a DCALF model decides its prediction rule according to the characteristics of the dataset. Execute Step 1 and Step 2 to obtain their decision graphs for rows and columns, as drawn in Figs. 6.6 and 6.7. Based on the decision graphs of Figs. 6.6(a) and 6.7(a), it can find that there are two cluster centers on D1.4 and three cluster centers on D2.4. In other words, D1.4 could be divided into two neighborhoods, while D2.4 is three. Therefore, DCALF model with Prediction Rule 1 would be adopted.

On the other hand, considering the decision graphs for columns as depicted in Figs. 6.6(b) and 6.7(b), there is only one cluster center on both D1.4 and D2.4. Namely, the columns involved do not have reliable neighborhoods. However, the columns have many outlier value, and marked by the red rectangles. Thus, Rule 4 is appropriate to implement prediction on these two datasets.

Table 6.1 Properties of all the designed test cases

Dataset	No.	Density (%)	Training data	Testing data
D1	D1.1	5	93,692	1,780,146
	D1.2	10	187,384	1,686,454
	D1.3	15	281,076	1,592,762
	D1.4	20	374,768	1,499,070
D2	D2.1	5	91,563	1,739,690
	D2.2	10	183,125	1,648,128
	D2.3	15	274,689	1,556,564
	D2.4	20	366,251	1,465,002

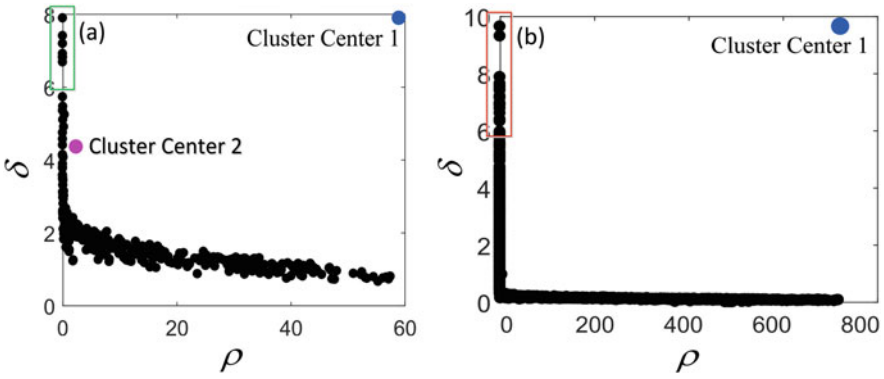


Fig. 6.6 Decision graph on D.1.4 in (a) users, (b) services

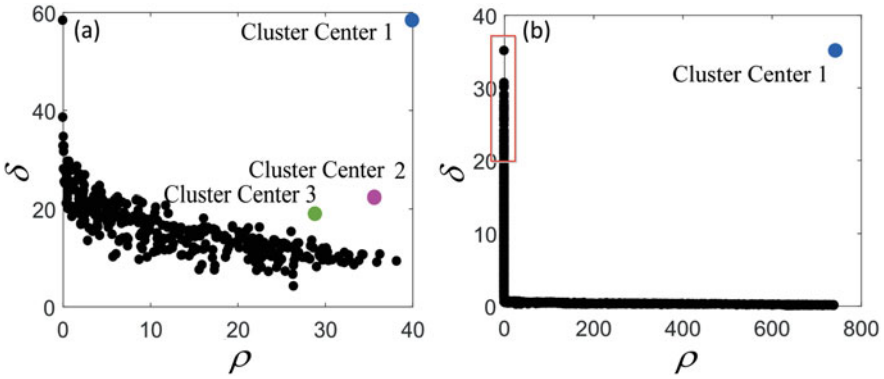


Fig. 6.7 Decision graph on D.2.4 in (a) users, (b) services

Note that the next experiments focus on validating the performance of DCALF models based on Prediction Rules 1 and 4. Hereafter, DCALF model with Prediction Rule 1 is renamed by DCALF-A, and DCALF model with Prediction Rule 4 is renamed by DCALF-B, respectively.

6.4.2 Performance Comparison

(a) Comparison of prediction accuracy
For comparative experiments and control variables, the latent feature dimension is set as $d = 20$ for all predictors except for AutoRec because it is a DNNs-based model. And the other parameters introduced in the compared predictors are set as their original article. DCALF model's other hyper-parameters are set as: the ratio

of unreliable data $\alpha = 0.2$ and $\eta = 0.01$ on D1, $\alpha = 0.1$ and $\eta = 0.0001$ on D2, $\lambda = 0.01$ and the average percentage of neighbors $P_{Vec} = 2\%$ on both D1 and D2.

Table 6.2 presents the MAE comparison results of all the models. To understand them easily, the win/loss counts of DCALF-A/DCALF-B with respect to other predictors are shown in Table 6.2. From the two tables, it can conclude that: (1) DCALF-A obviously implements lower MAE than other predictors on D2, but not on D1. (2) Because the unreliable services on D1 and D2 are obvious, it may be not appropriate to predict unobserved value based on neighborhoods of users (with DCALF-A), while predicting unobserved value based on reliable services (with DCALF-B)is highly doable.

To further explore the significance of the comparison results, the Friedman test is adopted in MAE and recorded in the last row of Table 6.2. From this results, the hypothesis "these comparison predictors have significant differences" is accepted at a significance level 0.05. The Friedman Rank value of DCALF-B is the lowest in all predictors, which means that DCALF-B has the highest prediction accuracy among its peers.

(b) Comparison of computational efficiency

The CPU running time of compared HDI data predictors on D1 and D2 is drawn in **Figs. 6.8 and 6.9**. It can find that: (1)the running time of BLF, RSNMF, and LN-LFM is less than other compared models, (2) the running time of DCALF is less than or comparable to NIMF, NAMF, LMF-PP, and GeoMF, and (3) the model with highest running time is AutoRec.

6.5 Summary

This chapter introduces a data-characteristic-aware latent factor (DCALF) model. It has the following towfold special designs: (1) it first extracts the dense latent features from the original raw HDI data by an LFL model, and (2) it employs DPClust method [21] to simultaneously identify the neighborhoods and outliers of HDI data on the extracted latent features. Experimental analyses show that DCALF significantly outperforms its competitors in missing data prediction of an HDI matrix. Besides, DCALF is also compatible with other side information, such as widely used geographic information. Hence, DCALF could be further improved by using other side information.

Table 6.2 Comparison results in MAE, including win/loss counts Statistic and Friedman Test, where ● and ✪ Respectively Indicate that DCALF-A and DCALF-B Have A Lower MAE than Comparison Models

Test									DCALF	
Cases	BLF	RSNMF	LN-LFM	NIMF	NAMF	GeoMF	LMF-PP	AutoRec	DCALF-A	DCALF-B
D1.1	0.5561●✪	0.5438✪	0.5501●✪	0.5502●✪	0.5465●✪	0.5305✪	0.5285✪	0.5467●✪	0.5457	0.5127
D1.2	0.4944●✪	0.4868●✪	0.4889●✪	0.4842✪	0.4976●✪	0.4827✪	0.4725✪	0.5055●✪	0.4857	0.4544
D1.3	0.4691●✪	0.4492✪	0.4611✪	0.4508✪	0.4625✪	0.4495✪	0.4472✪	0.4598✪	0.4642	0.4346
D1.4	0.4531●✪	0.4371✪	0.4424✪	0.4346✪	0.4360✪	0.4366✪	0.4260✪	0.4482✪	0.4520	0.4246
D2.1	18.9776●✪	21.4302●✪	18.6512●✪	17.7153●	20.2104●✪	24.7465●✪	18.3091●	21.3118●✪	17.6576	18.6237
D2.2	16.1924●✪	17.2305●✪	16.0634●✪	15.5264●✪	17.0126●✪	22.4728●✪	15.9125●✪	17.0310●✪	15.3595	15.3430
D2.3	14.9279●✪	14.6880●✪	14.7664●✪	14.2146✪	15.6547●✪	17.7908●✪	14.7450●✪	15.0156●✪	14.3836	14.0664
D2.4	14.3061●✪	14.3654●✪	14.1264●✪	13.5638✪	14.6482●✪	16.2852●✪	14.1033●✪	14.2265●✪	13.6697	13.5491
●win/loss	8/0	5/3	6/2	3/5	6/2	4/4	4/4	6/2	–	–
✪win/loss	8/0	8/0	8/0	7/1	8/0	8/0	7/1	8/0	–	–
Friedman rank*	8	6.125	6.25	3.75	7.375	6.875	3	7.625	4.625	**1.375**

a A lower Friedman Rank value indicates a higher prediction accuracy

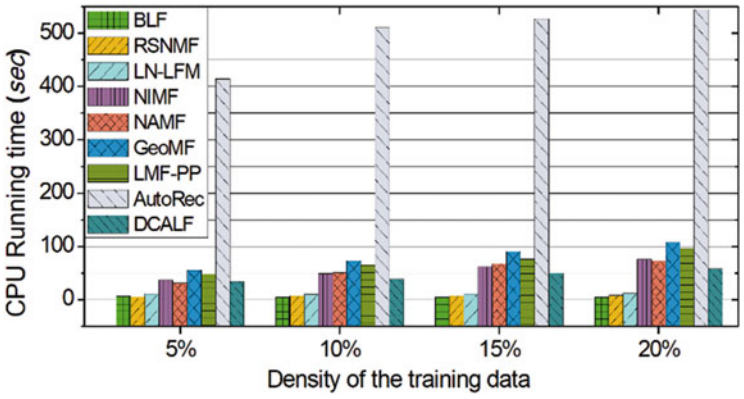


Fig. 6.8 CPU running time of comparison models on D1

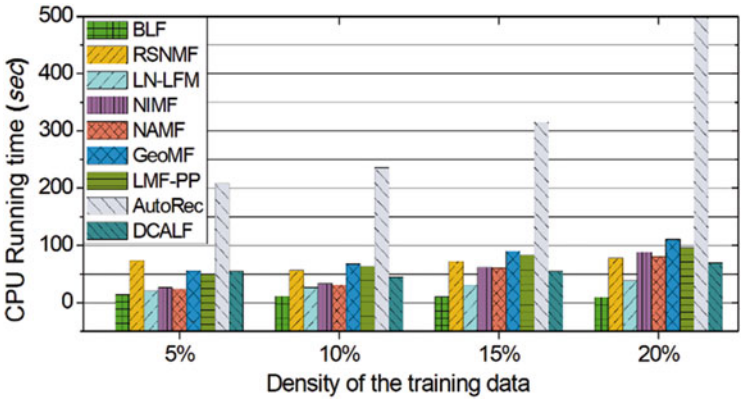


Fig. 6.9 CPU running time of comparison models on D2

References

1. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Wu, X.: A data-aware latent factor model for Web service QoS prediction. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 384–399 (2019)
2. Zhang, P., He, Y., Wu, D.: An ensemble latent factor model for highly accurate web service qos prediction. In: 2021 IEEE International Conference on Big Knowledge (ICBK), pp. 361–368 (2021)
3. Wu, D., Luo, X., He, Y., Zhou, M.: A prediction-sampling-based multilayer-structured latent factor model for accurate representation to high-dimensional and sparse data. In: IEEE Transactions on Neural Networks and Learning Systems, pp. 1–14 (2022)
4. Yang, Y., Zheng, Z., Niu, X., Tang, M., Lu, Y., Liao, X.: A location-based factorization machine model for web service QoS prediction. IEEE Trans. Serv. Comput. **14**(5), 1264–1277 (2018)
5. Feng, Y., Huang, B.: Cloud manufacturing service QoS prediction based on neighbourhood enhanced matrix factorization. J. Intell. Manuf. **31**(7), 1649–1660 (2020)
6. Luo, X., et al.: Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data. IEEE Trans. Cybern. **48**(4), 1216–1228 (2018). https://doi.org/10.1109/TCYB.2017.2685521
7. Luo, X., Zhou, M., Wang, Z., Xia, Y., Zhu, Q.: An effective scheme for QoS estimation via alternating direction method-based matrix factorization. IEEE Trans. Serv. Comput. **12**(4), 503–518 (2016)
8. Wu, D., Luo, X., Wang, G., Shang, M., Yuan, Y., Yan, H.: A highly accurate framework for self-labeled semisupervised classification in industrial applications. IEEE Trans. Industr. Inform. **14**(3), 909–920 (2017)
9. Luo, X., Zhou, M., Li, S., Wu, D., Liu, Z., Shang, M.: Algorithms of unconstrained non-negative latent factor analysis for recommender systems. IEEE Trans. Big Data. **7**(1), 227–240 (2019)
10. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Zhou, M.: A deep latent factor model for high-dimensional and sparse matrices in recommender systems. IEEE Trans. Syst. Man Cybern. Syst. **51**(7), 4285–4296 (2019)
11. Yuan, Y., Luo, X., Shang, M., Wu, D.: A generalized and fast-converging non-negative latent factor model for predicting user preferences in recommender systems. In: Proceedings of The Web Conference 2020, pp. 498–507 (2020)
12. Wu, D., Luo, X.: Robust latent factor analysis for precise representation of high-dimensional and sparse data. IEEE/CAA J. Autom. Sin. **8**(4), 796–805 (2021)
13. Luo, X., Liu, Z., Li, S., Shang, M., Wang, Z.: A fast non-negative latent factor model based on generalized momentum method. IEEE Trans. Syst. Man Cybern. Syst. **51**(1), 610–620 (2018)
14. Luo, X., Sun, J., Wang, Z., Li, S., Shang, M.: Symmetric and nonnegative latent factor models for undirected, high-dimensional, and sparse networks in industrial applications. IEEE Trans. Industr. Inform. **13**(6), 3098–3107 (2017)
15. Chen, Z., Shen, L., Li, F., You, D.: Your neighbors alleviate cold-start: on geographical neighborhood influence to collaborative web service QoS prediction. Knowl.-Based Syst. **138**, 188–201 (2017)
16. Wu, D., He, Q., Luo, X., Shang, M., He, Y., Wang, G.: A posterior-neighborhood-regularized latent factor model for highly accurate web service QoS prediction. IEEE Trans. Serv. Comput. **15**, 793 (2019)
17. Ludewig, M., Kamehkhosh, I., Landia, N., Jannach, D.: Effective nearest-neighbor music recommendations. In: Proceedings of the ACM Recommender Systems Challenge 2018, pp. 1–6 (2018)
18. Zhang, Y., Zhang, X., Zhang, P., Luo, J.: Credible and online qos prediction for services in unreliable cloud environment. In: 2020 IEEE International Conference on Services Computing (SCC), pp. 272–279 (2020)

19. Li, J., Wu, H., Chen, J., He, Q., Hsu, C.-H.: Topology-aware neural model for highly accurate QoS prediction. IEEE Trans. Parallel Distrib. Syst. **33**(7), 1538–1552 (2021)
20. Deng, S., Zhang, J., Wu, D., He, Y., Xie, X., Wu, X.: A quantitative risk assessment model for distribution cyber physical system under cyber attack. IEEE Trans. Industr. Inform., 1 (2022). https://doi.org/10.1109/TII.2022.3169456
21. Rodriguez, A., Laio, A.: Clustering by fast search and find of density peaks. Science. **344**(6191), 1492–1496 (2014)
22. Shin, H., Kim, S., Shin, J., Xiao, X.: Privacy enhanced matrix factorization for recommendation with local differential privacy. IEEE Trans. Knowl. Data Eng. **30**(9), 1770–1782 (2018)
23. He, Y., Wang, C., Jiang, C.: Correlated matrix factorization for recommendation with implicit feedback. IEEE Trans. Knowl. Data Eng. **31**(3), 451–464 (2018)
24. Wang, C. et al.: Confidence-aware matrix factorization for recommender systems. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1 (2018)
25. Wu, D., Jin, L., Luo, X.: PMLF: prediction-sampling-based multilayer-structured latent factor analysis. In: 2020 IEEE International Conference on Data Mining (ICDM), pp. 671–680 (2020)
26. Sun, B., Wu, D., Shang, M., He, Y.: Toward auto-learning hyperparameters for deep learning-based recommender systems. In: International Conference on Database Systems for Advanced Applications, pp. 323–331 (2022)
27. Wu, D., Zhang, P., He, Y., Luo, X.: A double-space and double-norm ensembled latent factor model for highly accurate web service QoS prediction. IEEE Trans. Serv. Comput., 1 (2022). https://doi.org/10.1109/TSC.2022.3178543
28. Shi, X., He, Q., Luo, X., Bai, Y., Shang, M.: Large-scale and scalable latent factor analysis via distributed alternative stochastic gradient descent for recommender systems. IEEE Trans. Big Data. **8**, 420 (2020)
29. Luo, X., Zhou, M., Xia, Y., Zhu, Q., Ammari, A.C., Alabdulwahab, A.: Generating highly accurate predictions for missing QoS data via aggregating nonnegative latent factor models. IEEE Trans. Neural Netw. Learn. Syst. **27**(3), 524–537 (2016). https://doi.org/10.1109/TNNLS.2015.2412037
30. Yu, D., Liu, Y., Xu, Y., Yin, Y.: Personalized QoS prediction for web services using latent factor models. In: 2014 IEEE International Conference on Services Computing, pp. 107–114 (2014)
31. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Collaborative web service QoS prediction via neighborhood integrated matrix factorization. IEEE Trans. Serv. Comput. **6**(3), 289–299 (2013). https://doi.org/10.1109/TSC.2011.59
32. Zhu, J., He, P., Zheng, Z., Lyu, M.R.: Online QoS prediction for runtime service adaptation via adaptive matrix factorization. IEEE Trans. Parallel Distrib. Syst. **28**(10), 2911–2924 (2017). https://doi.org/10.1109/TPDS.2017.2700796
33. Luo, X., Ouyang, Y., Xiong, Z.: Improving neighborhood based collaborative filtering via integrated folksonomy information. Pattern Recogn. Lett. **33**(3), 263–270 (2012)
34. Liu, Z., Luo, X., Wang, Z.: Convergence analysis of single latent factor-dependent, nonnegative, and multiplicative update-based nonnegative latent factor models. IEEE Trans. Neural Netw. Learn. Syst. **32**(4), 1737–1749 (2020)
35. Luo, X., Liu, Z., Jin, L., Zhou, Y., Zhou, M.: Symmetric nonnegative matrix factorization-based community detection models and their convergence analysis. IEEE Trans. Neural Netw. Learn. Syst. **33**(3), 1203 (2021)
36. Shi, X., Luo, X., Shang, M., Gu, L.: Long-term performance of collaborative filtering based recommenders in temporally evolving systems. Neurocomputing. **267**, 635–643 (2017)
37. Wu, D., Shang, M., Luo, X., Wang, Z.: An L_1-and-L_2-norm-oriented latent factor model for recommender systems. IEEE Trans. Neural Netw. Learn. Syst. **33**, 5775 (2021)
38. Wu, H., Luo, X., Zhou, M.: Advancing non-negative latent factorization of tensors with diversified regularizations. IEEE Trans. Serv. Comput. **15**(3), 1334 (2020)
39. Yuan, Y., Luo, X., Shang, M.-S.: Effects of preprocessing and training biases in latent factor models for recommender systems. Neurocomputing. **275**, 2019–2030 (2018)

40. Senthil Kumar, S., Margret Anouncia, S.: Qos-based concurrent user-service grouping for web service recommendation. Autom. Control. Comput. Sci. **52**(3), 220–230 (2018)
41. Tang, M., Zheng, Z., Kang, G., Liu, J., Yang, Y., Zhang, T.: Collaborative web service quality prediction via exploiting matrix factorization and network map. IEEE Trans. Netw. Serv. Manag. **13**(1), 126–137 (2016). https://doi.org/10.1109/TNSM.2016.2517097
42. Yin, Y., Aihua, S., Min, G., Yueshen, X., Shuoping, W.: QoS prediction for web service recommendation with network location-aware neighbor selection. Int. J. Softw. Eng. Knowl. Eng. **26**(04), 611–632 (2016)
43. Ryu, D., Lee, K., Baik, J.: Location-based web service QoS prediction via preference propagation to address cold start problem. IEEE Trans. Serv. Comput. **14**(3), 736–746 (2021). https://doi.org/10.1109/TSC.2018.2821686
44. Luo, X., Wu, H., Yuan, H., Zhou, M.: Temporal pattern-aware QoS prediction via biased non-negative latent factorization of tensors. IEEE Trans. Cybern. **50**(5), 1798–1809 (2020). https://doi.org/10.1109/TCYB.2019.2903736
45. Luo, X., Xia, Y., Zhu, Q.: Incremental collaborative filtering recommender based on regularized matrix factorization. Knowl.-Based Syst. **27**, 271–280 (2012)
46. Luo, X., et al.: An incremental-and-static-combined scheme for matrix-factorization-based collaborative filtering. IEEE Trans. Autom. Sci. Eng. **13**(1), 333–343 (2014)
47. Luo, X., Zhou, M., Li, S., You, Z., Xia, Y., Zhu, Q.: A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method. IEEE Trans. Neural Netw. Learn. Syst. **27**(3), 579–592 (2015)
48. Luo, X., Zhou, M., Li, S., Shang, M.: An inherently nonnegative latent factor model for high-dimensional and sparse matrices from industrial applications. IEEE Trans. Industr. Inform. **14**(5), 2011–2022 (2017)
49. Luo, X., Zhou, M., Xia, Y., Zhu, Q.: An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. IEEE Trans. Industr. Inform. **10**(2), 1273–1284 (2014)
50. Luo, X., Xia, Y., Zhu, Q.: Applying the learning rate adaptation to the matrix factorization based collaborative filtering. Knowl.-Based Syst. **37**, 154–164 (2013)
51. Luo, X., et al.: An efficient second-order approach to factorize sparse matrices in recommender systems. IEEE Trans. Industr. Inform. **11**(4), 946–956 (2015)
52. Luo, X., Zhou, M., Li, S., Hu, L., Shang, M.: Non-negativity constrained missing data estimation for high-dimensional and sparse matrices from industrial applications. IEEE Trans. Cybern. **50**(5), 1844–1855 (2020). https://doi.org/10.1109/TCYB.2019.2894283
53. Luo, X., Zhou, M., Shang, M., Li, S., Xia, Y.: A novel approach to extracting non-negative latent factors from non-negative big sparse matrices. IEEE Access. **4**, 2649–2655 (2016)
54. Shang, M., Luo, X., Liu, Z., Chen, J., Yuan, Y., Zhou, M.: Randomized latent factor model for high-dimensional and sparse matrices from industrial applications. IEEE/CAA J. Autom. Sin. **6**(1), 131–141 (2018)
55. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Wu, X.: A data-characteristic-aware latent factor model for web services QoS prediction. IEEE Trans. Knowl. Data Eng. **34**(6), 2525–2538 (2022)

Chapter 7
Posterior-neighborhood-regularized Latent Feature Learning

7.1 Overview

Quality-of-Service (QoS) is to measure the nonfunctional characteristics of web services [1–9]. If the QoS data of candidate web services is available, you can select and recommend Web services that meet the quality of service requirements of potential users. Warm-up tests that directly invoke Web services are often performed to retrieve QoS data [10, 11]. However, in real applications, the number of candidate services is usually large. Therefore, checking all candidate Web services is expensive, time-consuming, and therefore impractical [6, 12, 13].

Alternatively, another widely used approach for obtaining QoS data is QoS prediction [10, 12–16]. A QoS predictor is designed to accurately predict unknown QoS data based on historical ones. As a very successful implementation technology of e-commerce recommendation systems [17–28], Collaborative filtering (CF) has been widely used to develop a QoS predictor in recent years [29–34]. A CF-based QoS predictor is modeled based on a user-service QoS matrix [29–36], in which each column represents a web service, each row represents a user, and each entry represents the user's innovative QoS record of the service. The major problem of QoS prediction based on CF is to make accurate estimations based on the sparse user-service QoS matrix.

Among all kinds of CF-based QoS predictors [37–40], latent feature learning (LFL) is the most popular because of its scalability and accuracy [33, 38]. Neighborhood regularization is important for an LFL-based QoS predictor because similar users commonly experience similar QoS when invoking similar services [11, 14, 15, 33]. Most of the current neighborhood-regularized LFL-based QoS predictors identify the neighborhood of users'/services base on the common set. The common set is defined on the original local QoS matrix or additional geographical information. They rely on *prior neighborhood* and have the following major limitations:

1. It is difficult to accurately identify the neighbors of users/services only based on the common set defined on the extremely original sparse user-service QoS matrix.

D. Wu, *Robust Latent Feature Learning for Incomplete Big Data*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-19-8140-1_7

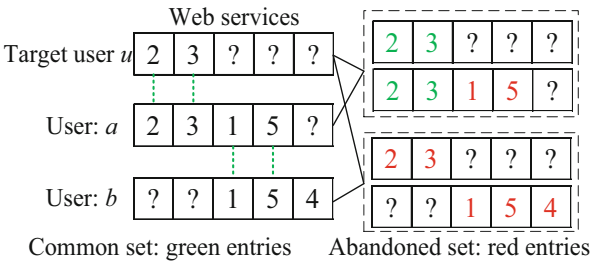


Fig. 7.1 The problem of building reliable neighborhoods on raw QoS data

As shown in Fig. 7.1, although target user u, user a, and user b are similar, user b is also considered to be a different user from the target user u because they do not have a common QoS record.

2. Due to issues such as identity privacy, information security and business interests, It is often difficult to retrieve the needed geographic information.

To overcome the above limitations, a *posterior-neighborhood*-regularized latent feature (PLF) model is proposed in [58]. PLF consists of three phases: primal latent feature extraction, posterior neighbor identification, and posterior-neighborhood-regularized LFL. As such, it can make accurate QoS predictions on raw QoS data.

7.2 Related Work

PLF is an LFL-based model [6, 17, 47]. An LFL model is built on a low-rank approximation of a sparse matrix based on its known entries only. An LFL-based QoS predictor maps both users/services into the same low-dimensional latent feature space to predict the missing entries [16, 29–31, 33, 38].

Neighborhood information can be used to improve the LFL-based QoS predictor's accuracy. Zheng et al. first proposed a neighborhood-integrated LFL-based QoS predictor [16]. Next, they integrated the network map of users [15]. After that, [14] incorporated the knowledge of geographical neighborhoods, [11] used the location information of users/services, and [33] systematically took into account geographical information, sample set diversity calculation, and platform environment. These proposed model all have neighborhood information, which defines neighborhood information as regular items.

In this study, PLF is significantly different from these models. First, some models [6, 17, 30, 38] usually suffer from low prediction accuracy because they ignore relationships between the users/services involved. Second, some models [11, 14, 15, 33] require additional geographical information to build neighborhood information, while PLF only depends on the user-service matrix.

In addition, Zhang et al. proposed a time-aware personalized QoS predictor based on a tensor factorization model [48] with the non-negative constraint [49]. Next,

some other time-aware models [50–57] are proposed. For PLF, it can also consider the time information.

7.3 A Posterior-Neighborhood-Regularized Latent Feature Model

Figure 7.2 shows the Flowchart of the PLF model. It consists of three phases: primal latent feature extraction, posterior neighbor identification, and posterior-neighborhood-regularized LFL.

7.3.1 Primal Latent Feature Extraction

A Euclidean distance-based objective function is a common choice for an LFL model [15, 17, 18]:

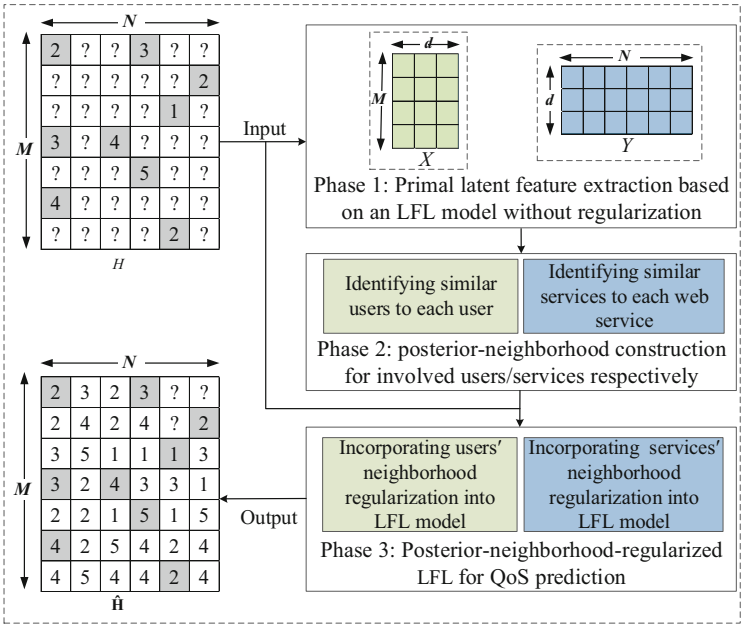


Fig. 7.2 Flowchart of the PLF model.

$$\arg\min_{X,Y} \varepsilon(X, Y) = \frac{1}{2} \sum_{(m,\ n)\in H_O} \left(h_{m,n} - \sum_{k=1}^{d} x_{m,k} y_{n,k} \right)^2 + \lambda \left(\|X\|_{L_2}^2 + \|Y\|_{L_2}^2 \right), \quad (7.1)$$

By considering the instant loss on a single instance $h_{m,n}$:

$$\varepsilon_{m,n} = \frac{1}{2} \left(h_{m,n} - \sum_{k=1}^{d} x_{m,k} y_{n,k} \right)^2. \quad (7.2)$$

To employ stochastic gradient descent (SGD) to minimize (7.2), it can obtain:

$$On\ h_{m,n}, for\ k = 1 \sim d : \begin{cases} x_{m,k} \leftarrow x_{m,k} + \eta y_{i,k} \left(h_{m,n} - \sum_{k=1}^{d} x_{m,k} y_{n,k} \right), \\ y_{n,k} \leftarrow y_{n,k} + \eta y_{n,k} \left(h_{m,n} - \sum_{k=1}^{d} x_{m,k} y_{n,k} \right). \end{cases} \quad (7.3)$$

7.3.2 *Posterior-Neighborhood Construction*

With the extracted primal latent feature matrices X and Y in Phase 1, the feature vectors of each user/service are obtained. In addition, by accurately fitting H_o, X and Y can well represent the information hidden in H. Thus, it can precisely identify target users'/services' posterior-neighbors based on X and Y. Figure 7.3 shows an example. Through the primal latent feature extraction, it can obtain the dense latent feature matrix X, which includes the feature vectors for target user u, user a, and user b. Based on X, it can identify users a and b as target user u's similar users. This addresses the limitation shown in Fig. 7.1.

For two users, p and j, computing their similarity $PCC_p(p, j)$ as follow

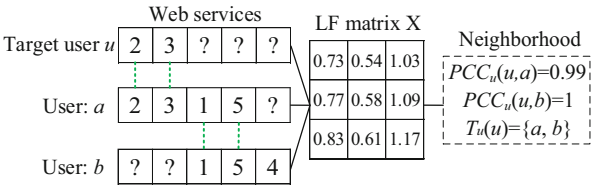


Fig. 7.3 An example of posterior-neighborhood construction.

$$\underset{p\in P, j\in P, p\neq j}{PCC_p(p,j)} = \frac{\sum_{k=1}^{d}\left(x_{p,k}-\overline{x}_p\right)\left(x_{j,k}-\overline{x}_j\right)}{\sqrt{\sum_{k=1}^{d}\left(x_{p,k}-\overline{x}_p\right)^2}\sqrt{\sum_{k=1}^{d}\left(x_{j,k}-\overline{x}_j\right)^2}}, \tag{7.4}$$

where $\overline{x}_p$and $\overline{x}_j$ are given respectively by:

$$\overline{x}_p = \frac{1}{d}\sum_{k=1}^{d} x_{p,k}, \overline{x}_j = \frac{1}{d}\sum_{k=1}^{d} x_{j,k}. \tag{7.5}$$

Note that $PCC_p(p,j)$ belongs to $[-1, 1]$, where a larger value represents a higher similarity. Based on user similarity given by (7.4), the K_1-nearest-neighbor set for each user can be identified. Then, the regularization neighbor set for each user can be obtained by

$$T_p(p) = \left\{j | j \in N_{K_1}(p),\ PCC_p(p,\ j) > 0,\ j \neq p\right\}. \tag{7.6}$$

Similarly, given two services i and j, their similarity $PCC_i(i, j)$ can be obtained by:

$$\begin{aligned} \underset{i\in N, j\in N, i\neq j}{PCC_i(i,j)} &= \frac{\sum_{k=1}^{d}\left(y_{i,k}-\overline{y}_i\right)\left(y_{j,k}-\overline{y}_j\right)}{\sqrt{\sum_{k=1}^{d}\left(y_{i,k}-\overline{y}_s\right)^2}\sqrt{\sum_{k=1}^{d}\left(y_{j,k}-\overline{y}_j\right)^2}}. \\ \overline{y}_i &= \frac{1}{d}\sum_{k=1}^{d} y_{i,k},\ \overline{y}_{jj} = \frac{1}{d}\sum_{k=1}^{d} y_{j,k}. \end{aligned} \tag{7.7}$$

The regularization neighbor set for each service can be obtained by:

$$T_i(i) = \{j | j \in N_{K_2}(i),\ PCC_i(i,\ j) > 0,\ j \neq i\}. \tag{7.8}$$

7.3.3 *Posterior-Neighborhood-Regularized LFL*

Since similar users tend to have similar QoS on similar services [14, 15], the following loss function can be considered:

$$\arg\min_{X',Y'} \varepsilon(X', Y') = \sum_{u=1}^{M}\left(\sum_{p\in T_u(u)} Su_{u,p}\sum_{k=1}^{d}\left(x\prime_{u,k}-x\prime_{p,k}\right)^2\right)+\sum_{i=1}^{N}\left(\sum_{j\in T_i(i)} Ss_{i,j}\sum_{k=1}^{d}\left(y\prime_{i,k}-y\prime_{j,k}\right)^2\right), \tag{7.9}$$

where $Su_{u,p}$ and $Ss_{i,j}$ model the linear weight of user u's p-th neighbor and service i's j-th neighbor, respectively, based on the corresponding posterior similarities.

$$Su_{u,p}=\frac{PCC_u(u,p)}{\sum_{p'\in T_u(u)} PCC_u(u,p')}, Ss_{i,j}=\frac{PCC_i(i,j)}{\sum_{j'\in T_i(i)} PCC_i(i,j')}. \tag{7.10}$$

Then, the posterior-neighborhood-regularized objective function is as follows:

$$\begin{aligned}\varepsilon(X', Y') = &\frac{1}{2}\sum_{(u,\ i)\in H_o}\left(h_{u,i}-\sum_{k=1}^{d} x\prime_{u,k}y\prime_{i,k}\right)^2+\frac{\lambda}{2}\sum_{(u,\ i)\in H_o}\left(\sum_{k=1}^{d}\left(x\prime_{u,k}\right)^2+\sum_{k=1}^{d}\left(y\prime_{i,k}\right)^2\right)\\ &+\frac{\alpha_1}{2}\sum_{u=1}^{M}\left(\sum_{p\in T_u(u)} Su_{u,p}\sum_{k=1}^{d}\left(x\prime_{u,k}-x\prime_{p,k}\right)^2\right)\\ &+\frac{\alpha_2}{2}\sum_{i=1}^{N}\left(\sum_{j\in T_i(i)} Ss_{i,j}\sum_{k=1}^{d}\left(y\prime_{i,k}-y\prime_{j,k}\right)^2\right),\end{aligned} \tag{7.11}$$

To train the instant loss, employing SGD:

$$On\ h_{u,i}, for\ k=1\sim d: \begin{cases} x'_{u,k}\leftarrow x'_{u,k}-\eta\dfrac{\partial\varepsilon_{u,i}}{\partial x'_{u,k}} \\ y'_{i,k}\leftarrow y'_{i,k}-\eta\dfrac{\partial\varepsilon_{u,i}}{\partial y'_{i,k}}\end{cases}. \tag{7.12}$$

Hence, the training rules of a PLF model are as follows:

$$On\ h_{u,i}, for\ k = 1 \sim d: \begin{cases} x'_{u,k} \leftarrow x'_{u,k} + \eta y'_{i,k}(h_{u,i} - x'_{u,k} y'_{i,k}) \\ \quad - \eta\lambda x'_{u,k} - \eta\alpha_1 \sum\limits_{p \in T_u(u)} Su_{u,p}\left(x'_{u,k} - x'_{p,k}\right), \\ y'_{i,k} \leftarrow y'_{i,k} + \eta x'_{u,k}(h_{u,i} - x'_{u,k} y'_{i,k}) \\ \quad - \eta\lambda y'_{i,k} - \eta\alpha_2 \sum\limits_{j \in T_i(i)} Ss_{i,j}\left(y'_{i,k} - y'_{j,k}\right). \end{cases} \tag{7.13}$$

Note that the time complexity of a PLF model is $\Theta(M^2 + N^2 + t_{max} \times (|H_O| + M \times K_1 + N \times K_2) \times d)$, where both t_{max} and d are positive constants [58].

7.4 Performance Analysis

7.4.1 General Settings

Datasets The adopted dataset is the Response Time, as described in Sect. 5.4.1. Table 7.1 summarizes the properties of all the test cases. In the experiments, the mean absolute error (MAE) and the root mean squared error (RMSE) are measured.

Baseline PLF is compared with several related models. The comparison models have different characteristics and are summarized in Table 7.2. In the experiments, $d = 20$ for all the models except for AutoRec to promote fair comparisons.

7.4.2 Comparisons Between PLF and State-of-the-Art Models

Figure 7.4 records the compared results. To validate whether PLF achieves significantly higher prediction accuracy than the other models, the Wilcoxon signed-ranks test [44] is applied to perform the statistical analysis. Table 7.3 records the statistical results. All the comparison results show that PLF has better QoS prediction accuracy than its competitors. Besides, Fig. 7.5 shows the CPU running time comparison on all testing cases. The results show that PLF's computational efficiency is comparable to its competitors.

Table 7.1 Properties of all the designed test cases

Dataset	No.	Density	Training data	Testing data
Response time	D1	5%	93,692	1,780,146
	D2	10%	187,384	1,686,454
	D3	15%	281,076	1,592,762
	D4	20%	374,768	1,499,070

Table 7.2 Descriptions of all the comparison models

Models	Descriptions
BMF	It is a basic matrix factorization (MF) model designed for recommendation [38].
NIMF	It is the neighborhood-integrated MF model by extending BMF with the information of similar users [16].
RSNMF	It is the regularized single element dependent non-negative MF model by applying regularization constraint [6].
NAMF	It is the network-aware MF model that employs the additional geographical information [15].
GeoMF	It improves BMF model by employing the additional geographical relationships [14].
LMF-PP	It is the location-based MF model that employs the additional geographical relationships [11].
AutoRec	It is deep neural networks-based model [41] designed for the recommendation, in which an autoencoder [42] is employed to implement CF.
PLF	The introduced model in this chapter [58].

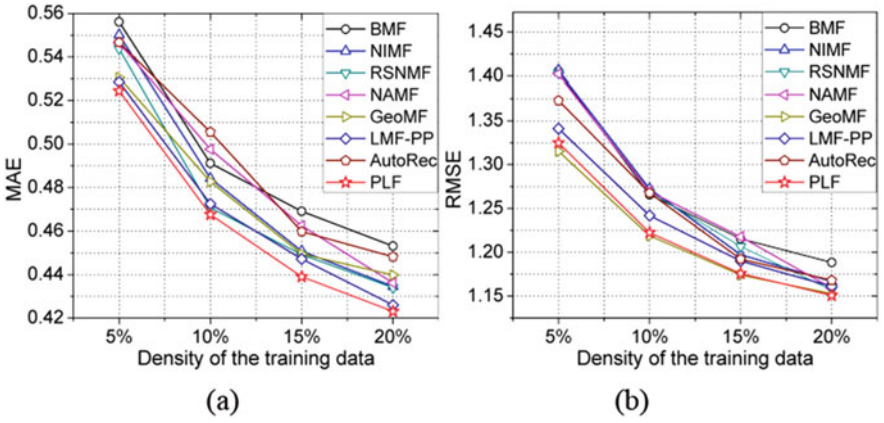


Fig. 7.4 The compared results on all the test cases: (a) MAE, (b) RMSE

Table 7.3 Statistical results of prediction accuracy by conducting Wilcoxon signed-ranks test with a significance level of 0.05

Comparison	R+	R-	p-value
PLF vs. BMF	36	0	0.0039
PLF vs. NIMF	36	0	0.0039
PLF vs. RSNMF	36	0	0.0039
PLF vs. NAMF	36	0	0.0039
PLF vs. GeoMF	27	9	0.1250
PLF vs. LMF-PP	36	0	0.0039
PLF vs. AutoRec	36	0	0.0039

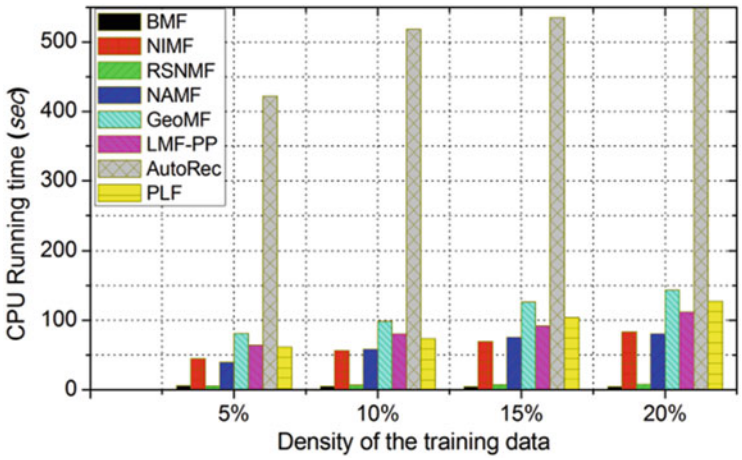


Fig. 7.5 The compared results of CPU running time on all the test cases

7.5 Summary

This chapter introduces a posterior-neighborhood-regularized latent feature (PLF) model. PLF model can achieve highly accurate Quality-of-Service (QoS) prediction in Web service recommendations. PLF has three phases: primal LF extraction, posterior neighbor identification, and posterior-neighborhood-regularized LFL. PLF is compared with several related models. The results show that PLF has better prediction accuracy than its competitors. Although PLF has shown promising prospects, several issues remain unveiled: (1) how to extend PLF to iteratively and dynamically identify users'/services' neighborhoods based on semi-supervised learning [44] or a deep-random-forest structure [43], and (2) it seems to be important in parallelling the PLF model [45, 46].

References

1. Luo, X., Zhou, M., Li, S., Wu, D., Liu, Z., Shang, M.: Algorithms of unconstrained non-negative latent factor analysis for recommender systems. IEEE Trans. Big Data. **7**(1), 227–240 (2021)
2. Wu, D., Shang, M., Wang, G., Li, L.: A self-training semi-supervised classification algorithm based on density peaks of data and differential evolution. In: 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC), pp. 1–6 (2018)
3. Zhang, P., He, Y., Wu, D.: An ensemble latent factor model for highly accurate web service qos prediction. In: 2021 IEEE International Conference on Big Knowledge (ICBK), pp. 361–368 (2021)
4. Purohit, L., Kumar, S.: A classification based web service selection approach. IEEE Trans. Serv. Comput. **14**, 315 (2018)

5. Lu, R., Jin, X., Zhang, S., Qiu, M., Wu, X.: A study on big knowledge and its engineering issues. IEEE Trans. Knowl. Data Eng. **31**(9), 1630–1644 (2019)
6. Luo, X., Zhou, M., Xia, Y., Zhu, Q., Ammari, A.C., Alabdulwahab, A.: Generating highly accurate predictions for missing QoS data via aggregating nonnegative latent factor models. IEEE Trans. Neural Netw. Learn. Syst. **27**(3), 524–537 (2016)
7. Luo, X., Yuan, Y., Wu, D.: Adaptive regularization-incorporated latent factor analysis. In: 2020 IEEE International Conference on Knowledge Graph (ICKG), pp. 481–488 (2020)
8. He, Y., Wu, D., Beyazit, E., Sun, X., Wu, X.: Supervised data synthesizing and evolving – a framework for real-world traffic crash severity classification. In: 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 163–170 (2018)
9. He, Y., Wu, B., Wu, D., Beyazit, E., Chen, S., Wu, X.: Toward mining capricious data streams: a generative approach. IEEE Trans. Neural Netw. Learn. Syst. **32**(3), 1228–1240 (2021)
10. Lee, K., Park, J., Baik, J.: Location-based web service QoS prediction via preference propagation for improving cold start problem. In: Proceeding of 2015 IEEE International Conference on Web Services, pp. 177–184 (2015)
11. Ryu, D., Lee, K., Baik, J.: Location-based web service QoS prediction via preference propagation to address cold start problem. IEEE Trans. Serv. Comput. **14**(3), 736–746 (2018)
12. Chen, X., Liu, X., Huang, Z., Sun, H.: Regionknn: a scalable hybrid collaborative filtering algorithm for personalized web service recommendation. In: Proceeding of 2010 IEEE International Conference on Web Services, pp. 9–16 (2010)
13. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Qos-aware web service recommendation by collaborative filtering. IEEE Trans. Serv. Comput. **4**(2), 140–152 (2011)
14. Chen, Z., Shen, L., Li, F., You, D.: Your neighbors alleviate cold-start: on geographical neighborhood influence to collaborative web service QoS prediction. Knowl.-Based Syst. **138**, 188–201 (2017)
15. Tang, M., Zheng, Z., Kang, G., Liu, J., Yang, Y., Zhang, T.: Collaborative web service quality prediction via exploiting matrix factorization and network map. IEEE Trans. Netw. Serv. Manag. **13**(1), 126–137 (2016)
16. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Collaborative web service qos prediction via neighborhood integrated matrix factorization. IEEE Trans. Serv. Comput. **6**(3), 289–299 (2013)
17. Luo, X., Zhou, M., Li, S., You, Z., Xia, Y., Zhu, Q.: A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method. IEEE Trans. Neural Netw. Learn. Syst. **27**(3), 579–592 (2016)
18. Shi, Y., Larson, M., Hanjalic, A.: Collaborative filtering beyond the user-item matrix: a survey of the state of the art and future challenges. ACM Comput. Surv. **47**(1), 1–45 (2014)
19. Chen, H., Li, J.: Learning multiple similarities of users and items in recommender systems. In: Proceeding of 2017 IEEE International Conference on Data Mining, pp. 811–816 (2017)
20. Zhao, H., Yao, Q., Kwok, J.T., Lee, D.L.: Collaborative filtering with social local models. In: Proceeding of 2017 IEEE International Conference on Data Mining, pp. 645–654 (2017)
21. Zhang, S., Wang, W., Ford, J., Makedon, F.: Learning from incomplete ratings using non-negative matrix factorization. In: Proceedings of 2006 SIAM International Conference on Data Mining, pp. 549–553 (2006)
22. Hernández-Lobato, J.M., Houlsby, N., Ghahramani, Z.: Probabilistic matrix factorization with non-random missing data. In: Proceedings of the 31st International Conference on Machine Learning, JMLR, pp. 1512–1520 (2014)
23. Wu, D., Shang, M., Luo, X., Wang, Z.: An L_1-and-L_2-norm-oriented latent factor model for recommender systems. In: IEEE Transactions on Neural Networks and Learning Systems, pp. 1–14 (2021)
24. Koren, Y., Bell, R.: Advances in collaborative filtering. In: Recommender Systems Handbook, pp. 77–118. Springer, Boston, MA (2015)
25. Chen, J., Wang, R., Wu, D., Luo, X.: A differential evolution-enhanced position-transitional approach to latent factor analysis. IEEE Transactions on Emerging Topics in Computational Intelligence, pp. 1–13 (2022)

26. Luo, X., Liu, Z., Li, S., Shang, M., Wang, Z.: A fast non-negative latent factor model based on generalized momentum method. IEEE Trans. Syst. Man Cybern. Syst. **51**, 610–620 (2018)
27. Wu, D., He, Y., Luo, X., Shang, M., Wu, X.: Online feature selection with capricious streaming features: a general framework. In: 2019 IEEE International Conference on Big Data (Big Data), pp. 683–688 (2019)
28. Wu, L., Sun, P., Hong, R., Ge, Y., Wang, M.: Collaborative neural social recommendation. IEEE Trans. Syst. Man Cybern. Syst. **51**, 464–476 (2018)
29. Wu, H., Yue, K., Li, B., Zhang, B., Hsu, C.-H.: Collaborative QoS prediction with context-sensitive matrix factorization. Futur. Gener. Comput. Syst. **82**, 669–678 (2018)
30. Zhu, J., He, P., Zheng, Z., Lyu, M.R.: Online qos prediction for runtime service adaptation via adaptive matrix factorization. IEEE Trans. Parallel Distrib. Syst. **28**(10), 2911–2924 (2017)
31. Wu, C., Qiu, W., Zheng, Z., Wang, X., Yang, X.: QoS prediction of web services based on two-phase k-means clustering. In: Proceeding of 2015 IEEE International Conference on Web Services, pp. 161–168 (2015)
32. Liu, A., Shen, X., Li, Z., Liu, G., Xu, J., Zhao, L., Zheng, K., Shang, S.: Differential private collaborative web services QoS prediction. World Wide Web. **22**, 2697–2720 (2018)
33. Feng, Y., Huang, B.: Cloud manufacturing service QoS prediction based on neighbourhood enhanced matrix factorization. J. Intell. Manuf. **31**, 1649–1660 (2018)
34. Luo, X., Zhou, M., Li, S., Xia, Y., You, Z.-H., Zhu, Q., Leung, H.: Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data. IEEE Trans. Cybern. **48**(4), 1216–1228 (2018)
35. Wang, M., Zheng, X., Yang, Y., Zhang, K.: Collaborative filtering with social exposure: a modular approach to social recommendation. In: Proceeding of the 33rd National Conference on Artifi-cial Intelligence, AAAI-18, pp. 1–8 (2017)
36. Yang, Y., Zheng, Z., Niu, X., Tang, M., Lu, Y., Liao, X.: A location-based factorization machine model for web service QoS prediction. IEEE Trans. Serv. Comput. **14**, 1264–1277 (2018)
37. Wu, J., Chen, L., Feng, Y., Zheng, Z., Zhou, M.C., Wu, Z.: Predicting quality of service for selection by neighborhood-based collaborative filtering. IEEE Trans. Syst. Man Cybern. Syst. **43**(2), 428–439 (2013)
38. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer. **42**(8), 30–37 (2009)
39. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Zhou, M.: A deep latent factor model for high-dimensional and sparse matrices in recommender systems. IEEE Trans. Syst. Man Cybern. Syst. **51**, 4285–4296 (2019)
40. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Qos-aware web service recommendation by collaborative filtering. IEEE Trans. Serv. Comput. **4**(2), 140–152 (2015)
41. Sedhain, S., Menon, A.K., Sanner, S., Xie, L.: AutoRec: autoencoders meet collaborative filtering. In: Proceedings of the 24th International Conference on World Wide Web, pp. 111–112 (2015)
42. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature. **521**(7553), 436–444 (2015)
43. Zhou, Z.-H., Feng, J.: Deep forest: towards an alternative to deep neural networks. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 3553–3559 (2017)
44. Wu, D., Luo, X., Wang, G., Shang, M., Yuan, Y., Yan, H.: A highly accurate framework for self-labeled semisupervised classification in industrial applications. IEEE Trans. Ind. Inform. **14**(3), 909–920 (2018)
45. Li, H., Li, K., An, J., Li, K.: MSGD: a novel matrix factorization approach for large-scale collaborative filtering recommender systems on GPUs. IEEE Trans. Parallel Distrib. Syst. **29**(7), 1530–1544 (2018)
46. Luo, X., Liu, H., Gou, G., Xia, Y., Zhu, Q.: A parallel matrix factorization based recommender by alternating stochastic gradient decent. Eng. Appl. Artif. Intell. **25**(7), 1403–1412 (2012)

47. Yao, L., Wang, X., Sheng, Q.Z., Benatallah, B., Huang, C.: Mashup recommendation by regularizing matrix factorization with API co-invocations. IEEE Trans. Serv. Comput. **14**(2), 502–515 (2018)
48. Zhang, Y., Zheng, Z., Lyu, M.R.: WSPred: a time-aware personalized QoS prediction framework for web services. In: Proceeding of the 22nd International Symposium on Software Reliability Engineering, pp. 210–219. IEEE (2011)
49. Zhang, W., Sun, H., Liu, X., Guo, X.: Temporal QoS-aware web service recommendation via non-negative tensor factorization. In: Proceedings of the 23rd International Conference on World Wide Web, pp. 585–596. ACM (2014)
50. Wang, X., Zhu, J., Zheng, Z., Song, W., Shen, Y., Lyu, M.R.: A spatial-temporal QoS prediction approach for time-aware web service recommendation. ACM Trans. Web. **10**(1), 1–25 (2016)
51. Wu, D., Luo, X., He, Y., Zhou, M.: A prediction-sampling-based multilayer-structured latent factor model for accurate representation to high-dimensional and sparse data. In: IEEE Transactions on Neural Networks and Learning Systems, pp. 1–14 (2022)
52. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Wu, X.: A data-characteristic-aware latent factor model for web service QoS prediction. IEEE Trans. Knowl. Data Eng. **34**(6), 2525–2538 (2022)
53. Luo, X., Chen, M., Wu, H., Liu, Z., Yuan, H., Zhou, M.: Adjusting learning depth in nonnegative latent factorization of tensors for accurately modeling temporal patterns in dynamic QoS data. IEEE Trans. Autom. Sci. Eng. **18**(4), 2142–2155 (2021)
54. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Wu, X.: A data-aware latent factor model for web service QoS prediction. In: Advances in Knowledge Discovery and Data Mining, pp. 384–399. Springer, Cham (2019)
55. Shang, M., Yuan, Y., Luo, X., Zhou, M.: An α-β-divergence-generalized recommender for highly accurate predictions of missing user preferences. IEEE Trans. Cybern. **52**, 8006–8018 (2021)
56. Luo, X., Wang, Z., Shang, M.: An instance-frequency-weighted regularization scheme for non-negative latent factor analysis on high-dimensional and sparse data. IEEE Trans. Syst. Man Cybern. Syst. **51**(6), 3522–3532 (2019)
57. Wu, D., Lu, G., Xu, Z.: Robust and accurate representation learning for high-dimensional and sparse matrices in recommender systems. In: 2020 IEEE International Conference on Knowledge Graph (ICKG), pp. 489–496 (2020)
58. Wu, D., He, Q., Luo, X., Shang, M., He, Y., Wang, G.: A posterior-Neighborhood-regularized latent factor model for highly accurate web service QoS prediction. IEEE Trans. Serv. Comput. **15**(2), 793–805 (2022)

Chapter 8
Generalized Deep Latent Feature Learning

8.1 Overview

In this era of big data, people are surrounding serious problems of information overload [1–9]. It is important to develop an intelligent system to filter the desired information from big data. In many real big-data-related applications, such as recommender systems and social networks [10–12], the collected data are commonly high dimensional and incomplete (HDI) [13–16]. Hence, the crux of implementing such an intelligent system is to effectively and efficiently analyze the HDI data.

Recently, deep learning has attracted much attention [17, 18]. It has been widely applied for various industrial applications such as face recognition, traffic control, natural language processing, pharmacy production, recommender systems, and social network analysis [40–42]. Since deep learning has a powerful representation learning ability and can mine higher-order data features, many researchers have applied deep learning to analyze HDI data [19–25]. However, most of them directly use the existing deep neural networks (DNNs) [26–32]. They did not fully consider the HDI characteristics of data, resulting in low computational efficiency and lack practicality [16]. Therefore, how to follow the principle of deep learning to improve the accuracy and computational efficiency of information granulation for HDI data remains to be studied.

To fill this gap, a deep latent feature (DLF) model was proposed in [33] for analyzing HDI data. Its main idea is to construct a latent feature learning (LFL)-based deep structure to enhance the LFL model's representation learning ability layer by layer. Note that DLF's deep structure is constructed by the LFL model rather than the DNNs. Hence, it has much higher computational and storage efficiency than DNNs-based models.

D. Wu, *Robust Latent Feature Learning for Incomplete Big Data*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-19-8140-1_8

8.2 Related Work

To date, many DNNs-based models have been proposed for analyzing HDI data [34–43]. The representative models include autoencoder-based [12], collaborative denoising autoencoder based [44], denoising autoencoder based [40], neural collaborative filtering-based [41], attentional factorization machines based [42], neural rating regression based [29], and deep collaborative conjunctive based [18] ones. Although DNNs-based model have great performance in HDI data analysis, they have some obvious limitations caused by the DNNs: (1) they require complete data of HDI data as inputs, which causes extremely high computational cost in some applications such as recommender systems, and (2) they have too many hyper-parameters that require very careful hyper-parameter tuning.

Following the principle of deep learning, Zhou and Feng proposed a Deep Forest [54] model to address the aforementioned limitations of DNNs. Its principle is to connect a series of decision trees to construct a deeply structured model. By doing this, the deep structure makes the resultant model also possess powerful representation learning ability with low computational and storage costs. According to the Deep Forest, it is possible to enhance a general learning model's representation learning ability by adopting the principle of the deep forest. From this point of view, a DLF model was proposed [33]. A DLF model's deep structure is constructed by connecting the LFL model layer by layer, enhancing representation learning ability with low computational and storage costs.

8.3 A Deep Latent Feature Model

Following the principle of a deep forest, DLF sequentially connects LFL models with different K layers and $K-1$ nonlinear activation functions. Given K $|M| \times d$ latent feature matrices $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_K\}$, K $d \times |K|$ latent feature matrices $\{\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_K\}$, and K $|M| \times |N|$ low-rank approximation matrices $\{\hat{\mathbf{H}}_1, \hat{\mathbf{H}}_2, \ldots, \hat{\mathbf{H}}_K\}$, DLF seeks for $\mathbf{X}_k$ and $\mathbf{Y}_k$ to obtain approximation $\hat{\mathbf{H}}_k$ for $\mathbf{H}$ first and then selects the best approximation $\hat{\mathbf{H}}$ as the final output, where $k \in \{1, 2, \ldots, K\}$. Figure 8.1 shows processing flow and structure of a DLF model. A DLF model works as follows:

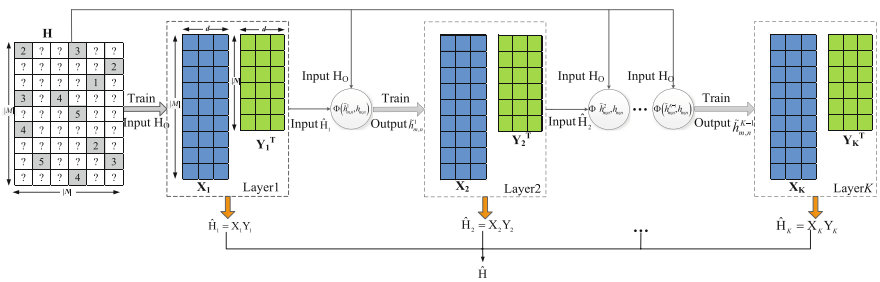


Fig. 8.1 The processing flow and structure of a DLF model

(a) H_O is taken as the initial input of the first layer;
(b) Training latent features matrices $\mathbf{X}_k$ and $\mathbf{Y}_k$ based on the input of the k-th layer and the learning rule to obtain $\hat{\mathbf{H}}_k$;
(c) Mixing the information in $\hat{\mathbf{H}}_k$ and $\mathbf{H}$ with a nonlinear activation function for enhancing the generality of a DLF model;
(d) The output of the activation function is taken as the input of the next layer; and.
(e) Repeating steps (b)–(d) to achieve $\{\hat{\mathbf{H}}_1, \hat{\mathbf{H}}_2, \ldots, \hat{\mathbf{H}}_K\}$, and the best approximation $\hat{\mathbf{H}}$ is selected from them as the final output.

Formally, the above processes are given by:

$$\arg\min_{\mathbf{X}_k, \mathbf{Y}_k} \varepsilon(\mathbf{X}_k, \mathbf{Y}_k) =$$
$$\begin{cases} \sum\limits_{h_{m,n} \in \mathbf{H}_O} \left(\left(h_{m,n} - \sum\limits_{f=1}^{d} x_{m,f}^1 y_{f,n}^1 \right)^2 + \lambda \left(\sum\limits_{f=1}^{d} \left(x_{m,f}^1 \right)^2 + \sum\limits_{f=1}^{d} \left(y_{f,n}^1 \right)^2 \right) \right), \text{if k} = 1; \\ \sum\limits_{h_{m,n} \in \mathbf{H}_O} \left(\left(\widetilde{h}_{m,n}^{k-1} - \sum\limits_{f=1}^{d} x_{m,f}^k y_{f,n}^k \right)^2 + \lambda \left(\sum\limits_{f=1}^{d} \left(x_{m,f}^k \right)^2 + \sum\limits_{f=1}^{d} \left(y_{f,n}^k \right)^2 \right) \right), \text{otherwise.} \end{cases} \tag{8.1}$$

where $xk\ m,f$ and $yk\ f,n$ are single latent features in $\mathbf{X}_k$ and $\mathbf{Y}_k$ with $k \in \{1, 2, \ldots, K\}$, and $\tilde{h}k\text{-}1\ m,n$ is the output of the nonlinear activation function designed as follows:

$$\begin{aligned} &\forall (m, n) \in \mathbf{H}_O, \forall n \in \{2, 3, \ldots, N\} : \\ &\widetilde{h}_{m,n}^{k-1} = \Phi\left(\widehat{h}_{m,n}^{k-1}, h_{m,n} \right) = \begin{cases} h_{m,n} & if\ \widehat{h}_{m,n}^{k-1} < h_{\min}, \\ h_{m,n} & if\ \widehat{h}_{m,n}^{k-1} > h_{\max}, \\ \widehat{h}_{m,n}^{k-1} & if\ \text{otherswise}; \end{cases} \end{aligned} \tag{8.2}$$

where $\hat{H}k\text{-}1\ m,n$ is a single element in $\hat{\mathbf{H}}_{k\text{-}1}$, and h_{min} and h_{max} denote the maximum and minimum values in $\mathbf{H}_O$, respectively. Note that if $\hat{H}k\text{-}1\ m,n < h_{min}$ or $\hat{H}k\text{-}1\ m, n > h_{max}$, the prediction from the k-1-th layer is obviously not correct. Thus, the function of (8.2) is to reset the extremely unreasonable ones predicted by the current layer. Then, considering the instant loss of (8.2) on each $h_{m,n}$ is as follows:

$$\varepsilon_{m,n}^{k}=\begin{cases}\left(h_{m,n}-\sum\limits_{f=1}^{d}x_{m,f}^{1}y_{f,n}^{1}\right)^{2}+\lambda\left(\sum\limits_{f=1}^{d}\left(x_{m,f}^{1}\right)^{2}+\sum\limits_{k=1}^{d}\left(y_{f,n}^{1}\right)^{2}\right), \text{ if } k=1;\\ \left(\widetilde{h}_{m,n}^{k-1}-\sum\limits_{f=1}^{d}x_{m,f}^{k}y_{f,n}^{k}\right)^{2}+\lambda\left(\sum\limits_{f=1}^{d}\left(x_{m,f}^{k}\right)^{2}+\sum\limits_{k=1}^{f}\left(y_{f,n}^{k}\right)^{2}\right), \text{ otherwise.}\end{cases} \tag{8.3}$$

Then, employing SGD to minimize (8.3) as follows:

$$\forall f\in\{1,2,\ \ldots,d\}:\begin{cases}x_{m,f}^{k}\leftarrow x_{m,f}^{k}-\eta\dfrac{\partial\varepsilon_{m,n}^{k}}{\partial x_{m,f}^{k}},\\ y_{f,n}^{k}\leftarrow y_{f,n}^{k}-\eta\dfrac{\partial\varepsilon_{m,n}^{k}}{\partial y_{f,n}^{k}},\end{cases} \tag{8.4}$$

where the stochastic gradients are given as follows:

$$\begin{aligned}&\text{if } k=1:\begin{cases}\dfrac{\partial\varepsilon_{m,n}^{1}}{\partial x_{m,f}^{1}}=-x_{m,f}^{1}\left(h_{m,n}-\widehat{h}_{m,n}^{1}\right)+\lambda x_{m,f}^{1}=-y_{f,n}^{1}\Delta_{m,n}^{1}+\lambda x_{m,f}^{1},\\ \dfrac{\partial\varepsilon_{m,n}^{1}}{\partial y_{f,n}^{1}}=-y_{f,n}^{1}\Delta_{m,n}^{1}+\lambda y_{f,n}^{1},\end{cases}\\ &\text{otherwise}:\begin{cases}\dfrac{\partial\varepsilon_{m,n}^{k}}{\partial x_{m,f}^{k}}=-y_{f,n}^{k}\left(\widetilde{h}_{m,n}^{k-1}-\widehat{h}_{m,n}^{k}\right)+\lambda x_{m,f}^{k}=-y_{f,n}^{k}\Delta_{m,n}^{k}+\lambda x_{m,f}^{k},\\ \dfrac{\partial\varepsilon_{m,n}^{k}}{\partial y_{f,n}^{k}}=-x_{m,f}^{k}\Delta_{m,n}^{k}+\lambda y_{f,n}^{k},\end{cases}\end{aligned} \tag{8.5}$$

where $\Delta k\ m,n$ denotes the prediction error between $h\ m,n$ and $\hat{H}k\ m,n$ or $\ddot{\tilde{h}}k\text{-}1\ m,n$ and $\hat{H}k\ m,n$. Thus, $\widehat{\mathbf{H}}_k$ can be obtained by:

$$\widehat{\mathbf{H}}_k=\mathbf{X}_k\mathbf{Y}_k. \tag{8.6}$$

Then, the generalized errors between $\mathbf{H}$ and $\widehat{\mathbf{H}}_k$ are denoted as E_k by:

$$E_k=\sum_{h_{m,n}\in\mathbf{H}_O}\left(h_{m,n}-\widehat{h}_{m,n}^{k}\right)^{2}. \tag{8.7}$$

The best approximation $\hat{\mathbf{H}}$ for $\mathbf{H}$ with the lowest generalized error E is selected as the final prediction as follows:

$$E \Leftarrow \min\{E_1, E_2, \ldots, E_N\} \tag{8.8}$$

As analyzed in [45–53], the time complexity of an LFL model is $\Theta(N_m \times |\mathbf{H}_O| \times d)$. Since the DLF model consists of K LFL models, the time complexity of a DLF model is K times than that of an LFL model. However, since an LFL is highly efficient in addressing the HDI data, the time complexity of a DLF model is acceptable in practice [48–50].

8.4 Performance Analysis

8.4.1 General Settings

Datasets Four benchmark HDI matrices are adopted as the datasets. 80% of known data of each dataset are used as the training set and the remaining 20% are used as the testing set. Five-fold cross-validations are adopted. Table 8.1 summarizes the adopted datasets.

Baselines Five related models are compared with the DLF model, including the original LFL model (i.e., BLF), non-negative matrix factorization based on regularized single element (RSNMF) [35], fast non-negative latent feature (FNLF) model [38], autoencoder based model (AutoRec) [12], and neural network-based matrix decomposition (NeuMF) [41]. Note that LFL, RSNMF, and FNLF are single-tier models, while NeuMF and AutoRec are DNNs-based models. RMSE and MAE are measured for all the models.

8.4.2 Effects of Layer Count in DLF

This set of experiments analyzes the performance of the DLF model regarding its layer count. The hyper-parameters are set as $\eta = 0.01$, $\lambda = 0.01$, $K = 10$, and $d = 20$,

Table 8.1 Properties of all the datasets

No.	Name	|M|	|N|	$\|H_o\|$	Density (%)
D1	Flixter	147,612	48,794	8,196,077	0.11
D2	Jester	24,983	100	1,186,324	47.49
D3	MovieLens_10M	71,567	65,133	10,000,054	0.21
D4	MovieLens_20M	138,493	26,744	20,000,263	0.54

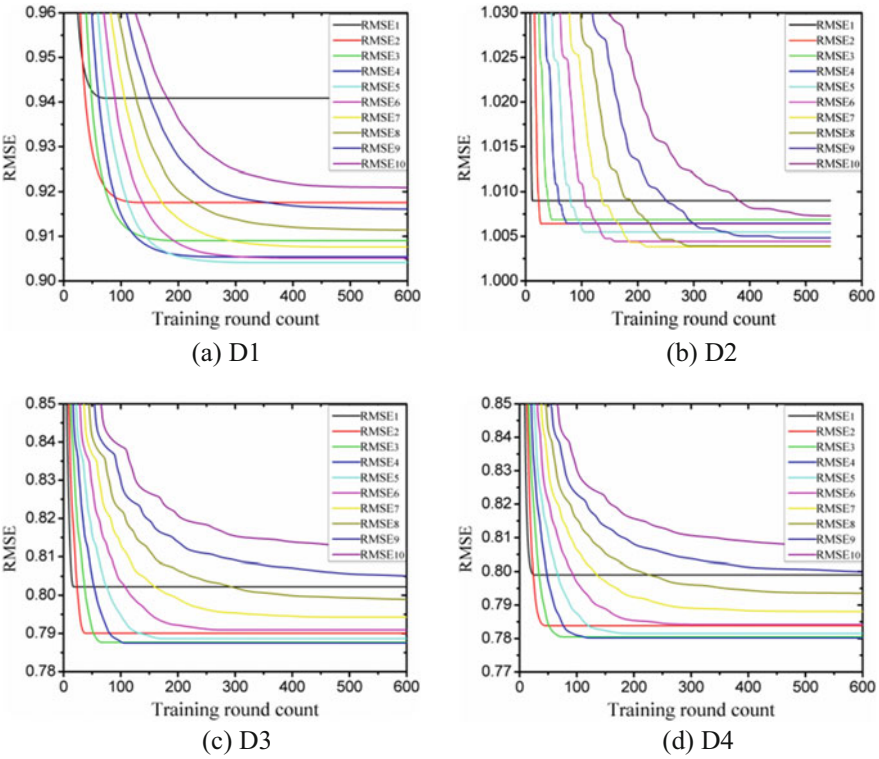


Fig. 8.2 RMSE training process of DLF at different layers

uniformly. Figures 8.2 and 8.3 record the training process of the DLF model at different layers. Figures 8.4 and 8.5 record the different RMSE and MAE of DLF as layer count changes. Since a DLF model degenerates into the original LFL model with only one layer, the results at the first layer are marked as the baseline. Then, the results can conclude the following important findings:

- RMSE/MAE keeps decreasing as training iteration increases at each layer. The deep design does not influence DLF's convergence at each layer.
- RMSE/MAE decreases first and then increases as layer count increases. The deep design can boost the LFL model's representation learning ability.

8.4.3 Comparison Between DLF and Related Models

Tables 8.2 and 8.3 record the comparison accuracy of all the models. Table 8.4 records the computational efficiency comparison. These results show that the DLF model achieves the lowest RMSE on D2, D3, and D4 and the lowest MAE on D2

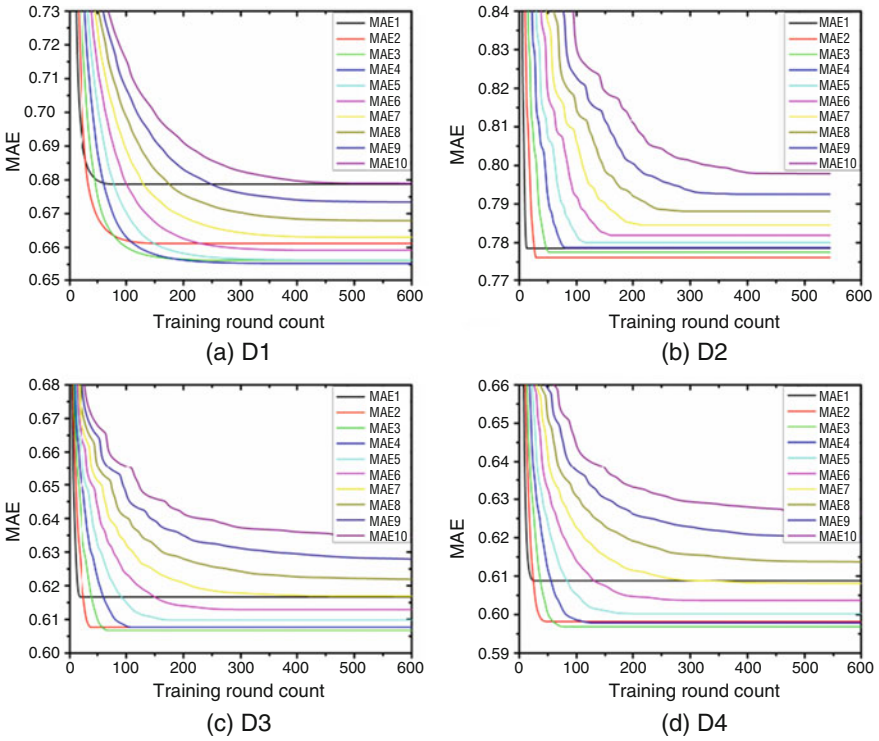


Fig. 8.3 MAE training process of DLF at different layers

and D3. In contrast, AutoRec has the lowest MAE on D1 and D4 and the lowest RMSE on D1. LFL, RSNMF, FNLF, and NeuMF do not achieve comparable prediction accuracy to DLF and AutoRec. The DLF model has higher computational efficiency than DNNs-based models.

8.5 Summary

This chapter introduces a deep latent feature (DLF) model for analyzing the HDI data. The DLF model's main idea is to construct an LFL-based deep structure to enhance the LFL model's representation learning ability layer by layer. Experimental results are conducted on four HDI datasets. The results demonstrate that the DLF model has comparable representation learning ability to and higher computational efficiency than DNNs-based models. In the future, a heterogeneous LFL model could be adopted to enhance the performance of the DLF model further.

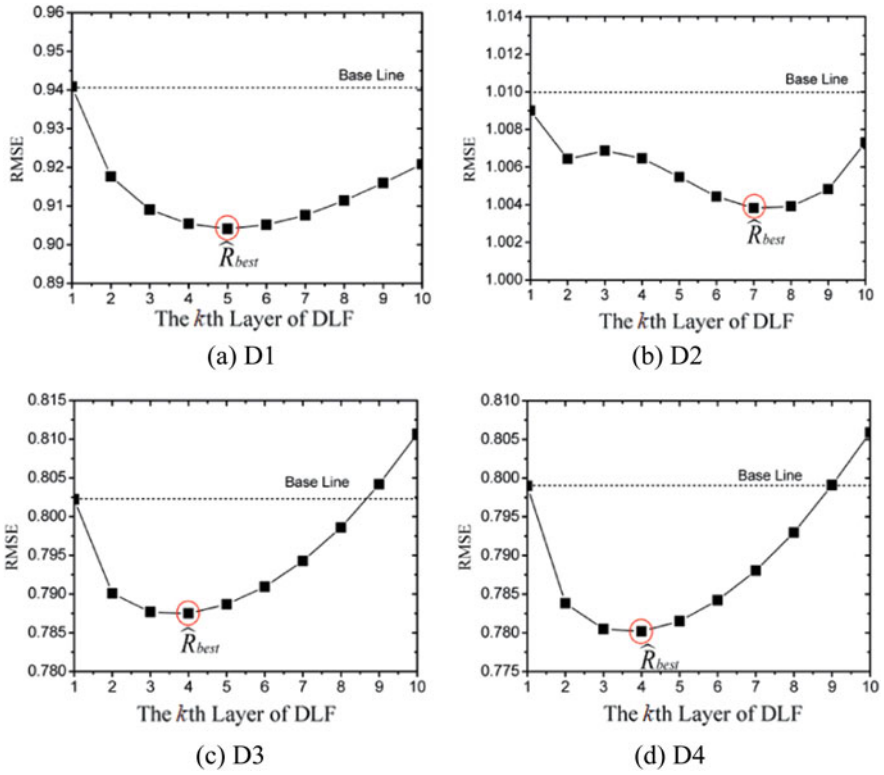


Fig. 8.4 Different RMSE of DLF as layer count changes

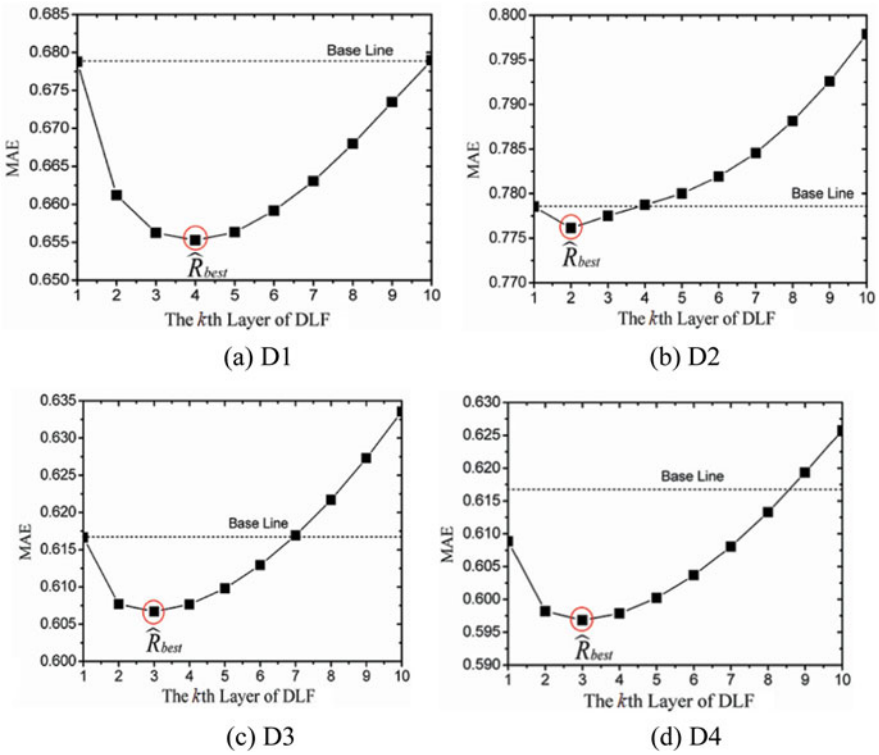


Fig. 8.5 Different MAE of DLF as layer count changes

Table 8.2 Lowest RMSE of all the models

Dataset	LFL	RSNMF	FNLF	AutoRec	NeuMF	DLF
D1	0.9224	0.9056	0.9038	0.8938	0.9174	0.9020
	($\lambda = 0.04$)	($\lambda = 0.05$)	($\lambda = 0.06$)	($\lambda = 100$)	($\lambda = 0.01$)	($\lambda = 0.004$)
D2	1.0087	1.0049	1.0025	1.0078	1.1047	1.0023
	(λ=0.02)	($\lambda = 0.08$)	($\lambda = 0.08$)	($\lambda = 100$)	(λ=0.01)	($\lambda = 0.004$)
D3	0.7981	0.7893	0.7881	0.7872	0.7977	0.7864
	(λ=0.03)	($\lambda = 0.05$)	($\lambda = 0.04$)	($\lambda = 100$)	(λ=0.01)	($\lambda = 0.006$)
D4	0.7935	0.7819	0.7798	0.7802	0.7922	0.7787
	(λ=0.03)	($\lambda = 0.05$)	($\lambda = 0.04$)	($\lambda = 100$)	(λ=0.01)	($\lambda = 0.004$)

Table 8.3 Lowest MAE of all the models

Dataset	LFL	RSNMF	FNLF	AutoRec	NeuMF	DLF
D1	0.6697	0.6550	0.6520	0.6472	0.6626	0.6537
	($\lambda = 0.03$)	($\lambda = 0.03$)	($\lambda = 0.04$)	($\lambda = 100$)	($\lambda = 0.01$)	($\lambda = 0.004$)
D2	0.7801	0.7769	0.7778	0.7905	0.7982	0.7750
	($\lambda = 0.01$)	($\lambda = 0.04$)	($\lambda = 0.03$)	($\lambda = 100$)	($\lambda = 0.01$)	($\lambda = 0.006$)
D3	0.6144	0.6080	0.6068	0.6062	0.6121	0.6053
	($\lambda = 0.02$)	($\lambda = 0.04$)	($\lambda = 0.03$)	($\lambda = 100$)	($\lambda = 0.01$)	($\lambda = 0.006$)
D4	0.6067	0.5977	0.5961	0.5947	0.6054	0.5956
	($\lambda = 0.02$)	($\lambda = 0.04$)	($\lambda = 0.03$)	($\lambda = 100$)	($\lambda = 0.01$)	($\lambda = 0.004$)

Table 8.4 Time cost (in seconds) of all the models

Dataset	LFL	RSNMF	FNLF	AutoRec	NeuMF	DLF
D1	58.78	406.94	900.12	5806.43	7014.32	1560.32
D2	19.65	35.67	10.3530	6990.14	5932.18	163.24
D3	56.23	442.21	902.87	50318.24	60267.25	549.42
D4	120.32	850.35	1305.48	60282.63	72168.56	1153.56

References

1. Shang, M., Yuan, Y., Luo, X., Zhou, M.: An α-β-divergence-generalized recommender for highly accurate predictions of missing user preferences. IEEE Trans. Cybern. **52**, 8006–8018 (2021)
2. Luo, X., Zhou, M., Li, S., Shang, M.: An inherently nonnegative latent factor model for high-dimensional and sparse matrices from industrial applications. IEEE Trans. Ind. Inform. **14**(5), 2011–2022 (2018)
3. You, Z.-H., Zhou, M., Luo, X., Li, S.: Highly efficient framework for predicting interactions between proteins. IEEE Trans. Cybern. **47**(3), 731–743 (2017)
4. Luo, X., Wu, H., Wang, Z., Wang, J., Meng, D.: A novel approach to large-scale dynamically weighted directed network representation. IEEE Trans. Pattern Anal. Mach. Intell., 1 (2021). https://doi.org/10.1109/TPAMI.2021.3132503
5. Zhang, J.-D., Chow, C.-Y., Xu, J.: Enabling kernel-based attribute-aware matrix factorization for rating prediction. IEEE Trans. Knowl. Data Eng. **29**(4), 798–812 (2017)
6. Wu, D., Zhang, P., He, Y., Luo, X.: A double-space and double-norm ensembled latent factor model for highly accurate web service QoS prediction. IEEE Trans. Serv. Comput., 1 (2022). https://doi.org/10.1109/TSC.2022.3178543
7. Li, Z., Li, S., Bamasag, O.O., Alhothali, A., Luo, X.: Diversified regularization enhanced training for effective manipulator calibration. IEEE Trans. Neural Netw. Learn. Syst., 1–13 (2022). https://doi.org/10.1109/TNNLS.2022.3153039
8. Gong, M., Jiang, X., Li, H., Tan, K.C.: Multiobjective sparse non-negative matrix factorization. IEEE Trans. Cybern. **49**(99), 1–14 (2018)
9. Wang, L., Gordon, M.D., Zhu, J.: Regularized least absolute deviations regression and an efficient algorithm for parameter tuning. In: Proceeding of the 8th IEEE International Conference on Data Mining (ICDM 2006), pp. 690–700
10. Wu, D., Luo, X.: Robust latent factor analysis for precise representation of high-dimensional and sparse data. IEEE/CAA J. Autom. Sin. **8**(4), 796–805 (2021)

11. He, Y., Wu, D., Beyazit, E., Sun, X., Wu, X.: Supervised data synthesizing and evolving – a framework for real-world traffic crash severity classification. In: Proceeding of 2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI, 2018), pp. 163–170
12. Sedhain, S., Menon, A.K., Sanner, S., Xie, L.: AutoRec: autoencoders meet collaborative filtering. In: Proceedings of the 24th International Conference on World Wide Web, pp. 111–112. ACM (2015)
13. Qi, Y., Jin, L., Luo, X., Zhou, M.: Recurrent neural dynamics models for perturbed nonstationary quadratic programs: a control-theoretical perspective. IEEE Trans. Neural Netw. Learn. Syst. **33**(3), 1216–1227 (2022)
14. Lu, H., Jin, L., Luo, X., Liao, B., Guo, D., Xiao, L.: RNN for solving perturbed time-varying underdetermined linear system with double bound limits on residual errors and state variables. IEEE Trans. Ind. Inform. **15**(11), 5931–5942 (2019)
15. Ryu, D., Lee, K., Baik, J.: Location-based web service QoS prediction via preference propagation to address cold start problem. IEEE Trans. Serv. Comput. **14**(3), 736–746 (2021)
16. Wu, D., Luo, X., He, Y., Zhou, M.: A prediction-sampling-based multilayer-structured latent factor model for accurate representation to high-dimensional and sparse data. IEEE Trans. Neural Netw. Learn. Syst., 1–14 (2022). https://doi.org/10.1109/TNNLS.2022.3200009
17. Deng, S., Zhang, J., Wu, D., He, Y., Xie, X., Wu, X.: A quantitative risk assessment model for distribution cyber physical system under cyber attack. IEEE Trans. Ind. Inform. (2022). https://doi.org/10.1109/TII.2022.3169456
18. Wang, Q., Peng, B., Shi, X., Shang, T., Shang, M.: DCCR: deep collaborative conjunctive recommender for rating prediction. IEEE Access. **7**, 60186–60198 (2019)
19. Luo, X., Chen, M., Wu, H., Liu, Z., Yuan, H., Zhou, M.: Adjusting learning depth in nonnegative latent factorization of tensors for accurately modeling temporal patterns in dynamic QoS data. IEEE Trans. Autom. Sci. Eng. **18**(4), 2142–2155 (2021)
20. Luo, X., Zhou, M., Li, S., Wu, D., Liu, Z., Shang, M.: Algorithms of unconstrained non-negative latent factor analysis for recommender systems. IEEE Trans. Big Data. **18**(4), 2142–2155 (2021)
21. Luo, X., et al.: An incremental-and-static-combined scheme for matrix-factorization-based collaborative filtering. IEEE Trans. Autom. Sci. Eng. **13**(1), 333–343 (2016)
22. Luo, X., Yuan, Y., Chen, S., Zeng, N., Wang, Z.: Position-transitional particle swarm optimization-incorporated latent factor analysis. IEEE Trans. Knowl. Data Eng. **34**, 3958 (2020)
23. Luo, X., Zhou, M.: Effects of extended stochastic gradient descent algorithms on improving latent factor-based recommender systems. IEEE Robot. Autom. Lett. **4**(2), 618–624 (2019)
24. Zhang, F., Jin, L., Luo, X.: Error-summation enhanced Newton algorithm for model predictive control of redundant manipulators. IEEE Trans. Ind. Electron., 1 (2022). https://doi.org/10.1109/TIE.2022.3165277
25. Sun, B., Wu, D., Shang, M., He, Y.: Toward auto-learning hyperparameters for deep learning-based recommender systems. In: Database Systems for Advanced Applications, pp. 323–331. Springer, Cham (2022)
26. He, Y., Wu, B., Wu, D., Beyazit, E., Chen, S., Wu, X.: Toward mining capricious data streams: a generative approach. IEEE Trans. Neural Netw. Learn. Syst. **32**(3), 1228–1240 (2021)
27. Wu, D., Luo, X., Wang, G., Shang, M., Yuan, Y., Yan, H.: A highly accurate framework for self-labeled semisupervised classification in industrial applications. IEEE Trans. Ind. Inform. **14**(3), 909–920 (2018)
28. Wu, D., He, Y., Luo, X., Zhou, M.: A latent factor analysis-based approach to online sparse streaming feature selection. IEEE Trans. Syst. Man Cybern. Syst. **52**, 6744–6758 (2021)
29. Li, P., Wang, Z., Ren, Z., Bing, L., Lam, W.: Neural rating regression with abstractive tips generation for recommendation. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 345–354 (2017)

30. Luo, X., Wang, Z., Shang, M.: An instance-frequency-weighted regularization scheme for non-negative latent factor analysis on high-dimensional and sparse data. IEEE Trans. Syst. Man Cybern. Syst. **51**(6), 3522–3532 (2019)
31. Wu, D., Shang, M., Luo, X., Wang, Z.: An L_1-and-L_2-norm-oriented latent factor model for recommender systems. IEEE Trans. Neural Netw. Learn. Syst. **33**, 5775–5788 (2021)
32. He, Y., Wu, B., Wu, D., Wu, X.: On partial multi-task learning. In: Proceeding of ECAI'2020, pp. 1174–1181 (2020)
33. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Zhou, M.: A deep latent factor model for high-dimensional and sparse matrices in recommender systems. IEEE Trans. Syst. Man Cybern. Syst. **51**(7), 4285–4296 (2021)
34. Luo, X., Zhou, M., Wang, Z., Xia, Y., Zhu, Q.: An effective scheme for QoS estimation via alternating direction method-based matrix factorization. IEEE Trans. Serv. Comput. **12**(4), 503–518 (2021)
35. Luo, X., Zhou, M., Xia, Y., Zhu, Q.: An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. IEEE Trans. Ind. Inform. **10**(2), 1273–1284 (2014)
36. Massa, P., Avesani, P.: Trust-aware recommender systems. In: Proceedings of the 2007 ACM Conference on Recommender systems, pp. 17–24. ACM (2007)
37. Brozovsky, L., Petricek, V.: Recommender system for online dating service. arXiv preprint cs/0703042 (2007)
38. Luo, X., Liu, Z., Li, S., Shang, M., Wang, Z.: A fast non-negative latent factor model based on generalized momentum method. IEEE Trans. Syst. Man Cybern. Syst. **51**(1), 610–620 (2021)
39. Luo, X., Wu, H., Yuan, H., Zhou, M.: Temporal pattern-aware QoS prediction via biased non-negative latent factorization of tensors. IEEE Trans. Cybern. **50**(5), 1798–1809 (2020)
40. Li, S., Kawale, J., Fu, Y.: Deep collaborative filtering via marginalized denoising auto-encoder.- In: Proceedings of the 24th ACM International Conference on Information and Knowledge Management, Melbourne, VIC, Australia, pp. 811–820 (2015)
41. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.-S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, pp. 173–182 (2017)
42. Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., Chua, T.-S.: Attentional factorization machines: learning the weight of feature interactions via attention networks. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI), pp. 3119–3125 (2017)
43. Luo, X., Sun, J., Wang, Z., Li, S., Shang, M.: Symmetric and nonnegative latent factor models for undirected, high-dimensional, and sparse networks in industrial applications. IEEE Trans. Ind. Inform. **13**(6), 3098–3107 (2017)
44. Wu, Y., DuBois, C., Zheng, A.X., Ester, M.: Collaborative denoising auto-encoders for top-N recommender systems. In: Proceedings of the 9th ACM International Conference on Web Search Data Mining, San Francisco, CA, USA, pp. 153–162 (2016)
45. Wu, D., He, Y., Luo, X., Shang, M., Wu, X.: Online feature selection with capricious streaming features: a general framework. In: Proceeding of 2019 IEEE International Conference on Big Data, pp. 683–688 (2019)
46. He, Y., Wu, B., Wu, D., Beyazit, E., Chen, S., Wu, X.: Online learning from capricious data streams: a generative approach. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence Main track (2019). https://doi.org/10.24963/ijcai.2019/346
47. Luo, X., Wang, D., Zhou, M., Yuan, H.: Latent factor-based recommenders relying on extended stochastic gradient descent algorithms. IEEE Trans. Syst. Man Cybern. Syst. **51**(2), 916–926 (2021)
48. Wu, D., Jin, L., Luo, X.: PMLF: prediction-sampling-based multilayer-structured latent factor analysis. In: 2020 IEEE International Conference on Data Mining (ICDM, 2020), pp. 671–680
49. Wu, D., Luo, X., Shang, M., He, Y., Wang, G., Wu, X.: A data-aware latent factor model for web service QoS prediction. In: Advances in Knowledge Discovery and Data Mining, pp. 384–399. Springer, Cham (2019)

50. Li, S., Zhou, M., Luo, X.: Modified primal-dual neural networks for motion control of redundant manipulators with dynamic rejection of harmonic noises. IEEE Trans. Neural Netw. Learn. Syst. **29**(10), 4791–4801 (2018)
51. Song, Y., Li, M., Zhu, Z., Yang, G., Luo, X.: Non-negative latent factor analysis-incorporated and feature-weighted fuzzy double c-means clustering for incomplete data. IEEE Trans. Fuzzy Syst. **30**, 4165–4176 (2022)
52. Xie, Z., Jin, L., Luo, X., Hu, B., Li, S.: An acceleration-level data-driven repetitive motion planning scheme for kinematic control of robots with unknown structure. IEEE Trans. Syst. Man Cybern. Syst. **52**, 5679 (2022)
53. Luo, X., Zhou, Y., Liu, Z., Hu, L., Zhou, M.: Generalized nesterov's acceler-ation-incorporated non-negative and adaptive latent factor analysis. IEEE Trans. Serv. Comput. **15**, 2809–2823 (2021)
54. Zhou, Z.-H., Feng, J.: Deep forest. Natl. Sci. Rev. **6**(1), 74–86 (2019)

Chapter 9
Conclusion and Outlook

9.1 Conclusion

Incomplete big data are frequently encountered in many industrial applications, such as recommender systems, the Internet of Things, intelligent transportation, bioinformatics, smart grid, online education, cloud computing, and so on. It is of great significance to analyze them for mining rich and valuable knowledge and patterns. Latent feature learning (LFL) is one of the most popular representation learning methods tailored for high dimensional and incomplete (HDI) data due to its high accuracy, computational efficiency, and ease of scalability. The crux of analyzing HDI data lies in addressing the uncertainty problem caused by their incomplete characteristics and some outliers (e.g., noises).

This book is aiming at introducing several robust latent feature learning methods to address such uncertainty for effectively and efficiently analyzing HDI data. First, it introduces the basic principle of LFL as well as two basic LFL models. Then, several robust LFL approaches are introduced, including robust latent feature learning based on smooth L_1-norm, improving robustness of latent feature learning using L_1-norm, improving robustness of latent feature learning using double-space, data-characteristic-aware latent feature learning, posterior-neighborhood-regularized latent feature learning, and generalized deep latent feature learning.

In general, this book can help readers understand how to employ LFL to build a robust model to analyze incomplete big data. In addition, this book provides several algorithms and real application cases, which can help students, researchers, and professionals easily build their models to analyze HDI data.

9.2 Outlook

There are several future research directions as follows.

D. Wu, *Robust Latent Feature Learning for Incomplete Big Data*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-981-19-8140-1_9

(a) Hyper-parameters tunning is time-consuming in LFL. It is possible to employ some intelligent optimization algorithms, e.g., differential evolution and particle swarm optimization, to make hyper-parameters tunning adaptive.
(b) By using deep learning techniques, it is possible to incorporate some side information, e.g., text and pictures, into LFL to improve its performance further.
(c) Since data privacy security has drawn more and more attention, it is possible to extend LFL to preserved privacy version by following the principle of federal learning.
(d) As LFL has excellent performance in mining rich and valuable knowledge and patterns from HDI data, it is possible to apply LFL to some specific HDI data analysis tasks.